Proceedings of the Workshop on Logic and Algorithms in Computational Linguistics 2017 (LACompLing2017)

Stockholm, August 16–19, 2017

Editors: Roussanka Loukanova and Kristina Liefke

Publisher: Stockholm: Stockholm University
October 30, 2017

Identifiers
URN: urn:nbn:se:su:diva-146800
OAI: oai:DiVA.org:su-146800
DiVA: diva2:1140018

Reviewers: Krasimir Angelov, Valeria de Paiva, Kristina Liefke, Roussanka Loukanova, Michael Moortgat, Reinhard Muskens

LACompLing2017 is a part of the series of events *Logic in Stockholm 2017*, August 7–25, 2017:
`https://www.math-stockholm.se/en/konferenser-och-akti/logic-in-stockholm-2`

LACompLing2017 is a workshop affiliated with the 26th EACSL Annual Conference on Computer Science Logic, CSL2017, August 20–24, 2017

Acknowledgement: LACompLing2017 is supported by a grant from the Swedish Research Council, Vetenskapsrådet, diarienummer: 2017-00571, to Roussanka Loukanova

Contents

Part I

Abstracts of Presentations and Short Biographies

Chris Barker New York University, New York, USA

Keynote Speaker

Parsing Sluicing

Abstract: Sluicing – common in English and cross-linguistically – involves anaphora to an incomplete constituent. For example, in "John called someone yesterday, but I don't know who @", the silent anaphoric element @ is interpreted as [John called __ yesterday], a constituent from which the indefinite "someone" has been scoped out (abstracted). I present a concrete substructural logic for sluicing that clarifies why parsing sluicing is more computationally costly than required by scope and anaphora alone.

Short Biography: Chris Barker has degrees in English literature, computer science, and theoretical linguistics. Since his 1991 Ph.D. at the University of California Santa Cruz, he has worked on possessives, vagueness, and scope. In 2014, he published an OUP monograph with Chung-chieh Shan, *Continuations in Natural Language*. Recent research topics include concealed questions, and negative polarity. He is Professor of Linguistics at New York University, where he serves as Vice Dean of the College of Arts and Science.

Wojciech Buszkowski Adam Mickiewicz University, Poland

Keynote Speaker

On Involutive Nonassociative Lambek Calculus

Abstract: Noncommutative Linear Logic is widely recognized as a logic closely related to type logics of categorial grammars, usually different versions of the Lambek calculus (Moot and Retoré 2012, Casadio and Lambek 2002, Morrill 1995). Here, we study a noncommutative and nonassociative linear logic, admitting two negations but no multiplicative constants. This logic is called Involutive Nonassociative Lambek Calculus (InNL), which follows the terminology in substructural logics. We focus on phase spaces for InNL and their applications, e.g. in a proof of cut elimination and P-TIME complexity. We also show that the languages generated by InNL-grammars are context-free.

Short Biography: Wojciech Buszkowski is Full Professor at the Faculty of Mathematics and Computer Science, Adam Mickiewicz University. He holds degree Dr. hab. in Mathematics (1988). In brief, his professional activities are represented as follows. Specialities: Mathematical Logic, Mathematical Linguistics, Computation Theory Research topics: Type grammars, Lambek calculi, Substructural logics, Learning algorithms, Knowledge representation, Modal logics, Tree automata Committees: a member of Editorial Board in Studia Logica, J. of Applied Logic, Logic J. of IGPL, Studies in Logic; a member of PC of several editions of LACL, MoL, FG. Associations: The Polish Association of Logic and Philosophy of Science (President 1993-1996); FoLLI (a member of The Committee of E.W. Beth Dissertation Award (2008-2016, chair 2010-2013)

Lucas Champollion New York University, USA

Invited Speaker

Two Switches in the Theory of Counterfactuals: A study of truth conditionality and minimal change (joint work with Ivano Ciardelli and Linmin Zhang)

Abstract: Based on a crowdsourced truth-value judgment experiment, we provide empirical evidence challenging two classical views in semantics, and we develop a novel account of counterfactuals that combines ideas from inquisitive semantics and causal reasoning. First, we show that two truth-conditionally equivalent clauses can make different semantic contributions when embedded

in a counterfactual antecedent. Assuming compositionality, this means that the meaning of these clauses is not fully determined by their truth conditions. This finding has a clear explanation in inquisitive semantics: truth-conditionally equivalent clauses may be associated with different propositional alternatives, each of which counts as a separate counterfactual assumption. Second, we show that our results contradict the common idea that the interpretation of a counterfactual involves minimizing change with respect to the actual state of affairs. Building on techniques from causal reasoning, we propose to replace the idea of minimal change by a distinction between foreground and background for a given counterfactual assumption: the background is held fixed in the counterfactual situation, while the foreground can be varied without any minimality constraint. (Joint work with Ivano Ciardelli and Linmin Zhang)

SHORT BIOGRAPHY: Lucas Champollion is Assistant Professor of Linguistics at New York University. He holds a Ph.D. in Linguistics (2010) and a M.Sc. in Computer and Information Science (2007), both from the University of Pennsylvania. Before joining New York University in 2012, he worked as a postdoctoral researcher at Eberhard Karls Universität Tübingen in Germany. His work has appeared in journals such as *Theoretical Linguistics*, *Linguistics and Philosophy*, the *Journal of Semantics*, as well as *Semantics and Pragmatics*.

Robin Cooper University of Gothenburg, Sweden

Keynote Speaker

Neural TTR

ABSTRACT: One of the claims of TTR (Type Theory with Records) is that it can be used to model types learned by agents in order to classify objects and events in the world, including speech events. That is, the types can be represented by patterns of neural activation in the brain. Of course, this claim would be empty if it turns out that the types are in principle impossible to represent on a finite network of neurons. In this talk, we will discuss how to represent types in terms of neural events on a network and present a preliminary implementation that maps types to events on a network. The kind of networks we will use are closely related to the transparent neural networks (TNN) discussed by Strannegård. In particular, we will discuss:

1. the binding problem – making sure one can distinguish between the type of events where a hugs b, $hug(a, b)$, and the type of events where b hugs a, $hug(b, a)$.
2. the recursion problem – making sure that one can allow for types to be embedded within types to an arbitrary depth: $believe(c, hug(a, b))$, $know(d, believe(c, hug(a, b)))$, and so on.
3. dealing with quantification – our solution involves generalized quantifiers rather than quantifiers from first-order logic.
4. memory – if a type corresponds to an event on a neural system, rather than an architectural feature of the neural system, as seems necessary to handle 1–3, what can it mean to have a memory that some object/event is of a type? Our proposed solution uses an idea from TNN where a single neuron can, when activated, trigger an occurrence of the neural event corresponding to a type judgement.

The discussion will be based on a preliminary Python implementation using the Python implementation of TTR (pyttr).

SHORT BIOGRAPHY: Robin Cooper is Senior Professor at the University of Gothenburg, where he was previously Professor of Computational Linguistics.

He is currently conducting research within the Centre for Linguistic Theory and Studies in Probability (CLASP) at Gothenburg. He has an undergraduate degree from the University of Cambridge and a Ph.D. in Linguistics from the University of Massachusetts at Amherst. He has taught at the following universities: Universität Freiburg, University of Texas at Austin, University of Massachusetts at Amherst, University of Wisconsin at Madison, Stanford University, Lund University, and Edinburgh University. He has held a Mellon Postdoctoral Fellowship and a Guggenheim Fellowship, and has been a fellow at the Center for Advanced Study in the Behavioral Sciences at Stanford. He is a Fellow of the British Academy and the Royal Society of Arts and Sciences in Gothenburg and a member of Academia Europaea. He holds an honorary doctorate from Uppsala University. His main research interests are semantics (both theoretical and computational), dialogue semantics and computational dialogue systems. He is currently working on a type-theoretical approach to language and cognition.

Ann Copestake University of Cambridge, UK

Keynote Speaker

Logical Forms in Broad-Coverage Grammar Development

Abstract: I will outline some of the work on compositional semantics with large-scale computational grammars and in particular work using Minimal Recursion Semantics (MRS) in DELPH-IN. There are grammar fragments for which MRS can be converted into a logical form with a model-theoretic interpretation, but I will argue that attempting to use model theory to justify the MRS structures in general is inconsistent with the goals of grammar engineering. I will outline some alternative approaches to integrating distributional semantics with this framework and show that this also causes theoretical difficulties. In both cases, we could consider inferentialism as an alternative theoretical grounding whereby classical logical properties are treated as secondary rather than primary. In this view, it is important that our approaches to compositional and lexical semantics are consistent with uses of language in logical reasoning, but it is not necessary to try and reduce all aspects of semantics to logic.

Short Biography: Ann Copestake started doing research in Computational Linguistics at the University of Cambridge in 1985. From 1994 to 2000, she worked at CSLI, Stanford, where she became part of the LinGO project, building broad-coverage grammars using HPSG. She returned to Cambridge in 2000 and is now a Professor there.

Henriëtte de Swart Utrecht Institute of Linguistics OTS - Language, logic and information, The Netherlands

Keynote Speaker

Time in Translation

Abstract: Linguists and philosophers have long been interested in ways languages refer to time. With today's large corpora, computational linguists have started to apply these insights to improve automatic translation programmes, but that proves to be quite a challenge. Purely statistical approaches are insufficient: languages like English, Dutch, German, French and Spanish all have past (e.g. 'broke'), present ('breaks') and perfect ('has broken') verb forms in their grammar, but they don't use them under the same (sentential/discourse) conditions. The perfect is the bottleneck: 'has broken' reports on a past event

of breaking that has current relevance, so it shares features of the past as well as the present tense.

We hypothesize that the use of the perfect in a particular language depends on how the grammar distributes past and present meanings over the various verb forms. Accordingly, we do not study the perfect in isolation, but focus on its competition with past and present. Instead of avoiding the cross-linguistic variation, we embrace it to unveil the meaning of the perfect through parallel corpus research (Europarl, Open Subtitles, literary translations).

Translation equivalents provide us with form variation across languages in contexts where the meaning is stable. Our method of Translation Mining extracts verb forms from one language and aligns them with verb forms in other languages. Language specific tense annotations constitute the input to a dissimilarity matrix, so we can use multidimensional scaling to draw temporal maps. A user-friendly interface that connects the maps to the underlying data reveals how tense choice is sensitive to lexical semantics, compositional semantics, discourse and pragmatics. The talk will present early results that show how our data challenge established ideas in linguistic theory and generate new hypotheses.

In this way, computational techniques enable us to ground linguistic theory in a usage-based approach. Beyond temporal reference, the investigation of meaning in translation provides a new methodology for cross-linguistic semantic research.

Short Biography: After she obtained her Ph.D. at Groningen University in 1991, Henriette de Swart taught at Groningen University and Stanford University until she became a full professor in French linguistics and semantics at Utrecht University. Her research focuses on cross-linguistic variation in meaning. She published journal articles and books/book chapters on tense and aspect, negation, bare nominals and indefinites, as well as an introductory textbook in semantics. She also investigated the role of semantics in language evolution, and was closely involved in the development of bidirectional optimality theory. She teaches courses in the bachelor programme French language and culture, the bachelor programme Artificial Intelligence, and the research master Linguistics. She has been director of the national graduate school in linguistics (LOT) and the director of the Utrecht Institute of Linguistics (LOT). Since 2013, she is a member of the Royal Netherlands Academy of Sciences (KNAW).

Fredrik Engström University of Gothenburg, Sweden

Invited Speaker

Generalized Quantifiers and Team Semantics

Abstract: Dependence logic is an elegant way of introducing dependencies between variables into the object language. Generalized quantifiers can be introduced in this setting, but a satisfying account can only be given for monotone increasing generalized quantifiers. In this talk I will build a new logic on the same framework as dependence logic, team semantics, that allows handling of any generalized quantifier in a natural way. I will show how to interpret Dependence logic into the logic and give a characterization of its logical strength.

Short Biography: Fredrik Engstrm is a researcher of logic at the Department of Philosophy, Linguistics and Theory of science of the University of Gothenburg. He did his Ph.D. in Mathematics at the University of Birmingham and Chalmers University of Technology in 2004 with a thesis on mathematical logic and models of Peano's arithmetic, under supervision of Richard W. Kaye.

After which he worked as an associate professor at the Mid Sweden University in Sundsvall before starting at the University of Gothenburg in 2006. His research is now mainly focused on Dependence logic and the framework of team semantics that underlies the development of logics that are able to express dependences between variables.

Stefan Evert, Philipp Heinrich, Klaus Henselmann, Ulrich Rabenstein, Elisabeth Scherr, and Lutz Schröder FAU Erlangen-Nürnberg

Combining Machine Learning and Semantic Features in the Classification of Corporate Disclosures

ABSTRACT: We investigate an approach of improving statistical text classification by combining machine learners with an ontology-based identification of domain-specific topic categories. We apply this approach to *ad hoc* disclosures by public companies. This form of obligatory publicity concerns all information that might affect the stock price; relevant topic categories are governed by stringent regulations. Our goal is to classify disclosures according to their effect on stock prices (negative, neutral, positive). In the feasibility study reported here, we combine natural language parsing with a formal background ontology to recognize disclosures concerning a particular topic, viz. retirement of key personnel. The semantic analysis identifies such disclosures with high precision and recall. We then demonstrate that machine learners benefit from the additional ontology-based information in different prediction tasks.

Gintare Grigonyte Stockholm University, Sweden

Invited Speaker

Hybrid Way. Experience from organising domain language into knowledge bases, language acquisition and language evolution

ABSTRACT: The unifying perspective of this talk is hybridisation between corpus linguistics and computational linguistics. The first facet of the talk discerns how corpus linguistics' methods can be wielded for making the knowledge in raw text accessible; and organizing it for specific domains like computer science or medicine in terms of terminology extraction and tracking language change. The second facet of the talk is about how methods from computational linguistics can extend corpus linguistics research in areas like first and second language acquisition.

SHORT BIOGRAPHY: Gintare Grigonyte is a researcher at the Department of Linguistics, Stockholm University. Her research aims at providing insight into the nature and structure of natural human language by applying computational techniques. Her research up to date has focused on several topics: automatic extraction of domain terminologies and semantic relationships, multilingual lexicography through information retrieval, biomedical NLP and language evolution, and NLP for first and second language acquisition.

Justyna Grudzińska and Marek Zawadowski University of Warsaw, Poland

Relational Nouns, Inverse Linking and Haddock Descriptions: A unified dependent type account

ABSTRACT: In this talk, we argue for a unified dependent type analysis of two puzzling phenomena: inverse linking and Haddock descriptions. Inverse linking constructions (ILCs) refer to the syntactic environments in which the embedded quantifier phrase (QP) takes scope over the embedding one. For

example, sentence (1) *A representative of every country missed a meeting* can be understood to mean that a different representative of each country missed a potentially different meeting in each case, i.e., it allows a reading in which *every country* outscopes *a representative*. This poses a puzzle for standard scope-assignment strategies, for there is independent evidence that scoping out of DP islands should be disallowed. For example, sentence (2) *Two politicians spy on someone from every city* cannot be understood to mean that for every city c, there are two politicians who spy on someone from c, i.e., it does not have a reading in which *two politicians* takes scope in between the two nested QPs. The second phenomenon to be discussed in our talk relates to the so-called Haddock descriptions. Haddock (relative) descriptions, e.g. *the rabbit in the hat*, pose a difficulty for standard presuppositional accounts of definite descriptions, for they carry a kind of 'polyadic presupposition' that there is a single pair $\langle x, y\rangle$ such that x is a hat, y is a rabbit, and y is in x, rather than the standard presuppositions that there be a unique rabbit and a unique hat. Thus *the rabbit in the hat* can be used felicitously in the context with multiple salient hats and multiple salient rabbits, as long as there is exactly one pair of a rabbit and a hat such that the rabbit is in the hat.

Lars Hellan Norwegian University of Science and Technology, Norway

Keynote Speaker

Large Scale 'Deep' Grammars — a success, and a long way to go

Abstract: Grammars with more or less complete lexical and morpho-syntactic coverage now exist, with mapping to semantic represenations akin to representations in standard predicate logic (like MRS). With the ability to design such grammars, the goals of as well classical generative grammar as Montague grammar may be said to have been realized, an achievement also not possible without the various types of computational backbones sustaining these grammars. Multi-word expressions, a longstanding 'black hole' of structurally-driven grammar modeling, are now getting due attention, helping to consolidate the grammars as 'grammars of the language'. Dimensions of further research and development relate to the circumstance that every large scale grammar is linked to its developer, and to the language represented. The latter link manifests itself in two facts, for a given pair of languages L1 and L2 : (i) There is no facility yet for systematically reading off from their grammars what L1 and L2 have in common (or relate to each other in any larger perspective). (ii) In the (rather normal) case where L1 and L2 have different syntax and different strategies of lexical encoding, semantically equivalent expressions in L1 and L2 yet do not receive the same representation (MRS, for instance, reflecting the wording of the language). The link to the developer is not personal, but resides in the circumstance that a grammar of such a complexity is the product of a totality of solutions to a very large number of parameters, a totality that most likely can be traced to only one person. How can such a system be sustainable? A solution to the latter link may possibly be found only if we make progress on the former link.

Short Biography: Having been 'raised' in generative grammar and formal semantics, Hellan has invested some effort over the last decades in the construction of a large Norwegian HPSG grammar. At the same time having been involved in typological and documentary projects especially in Africa, he has been interested in the development of resources for facilitating systematic comparative studies and language particular resource building, such as valency

databases.

Valia Kordoni Humboldt University Berlin, Germany

Keynote Speaker

Multiword Expressions and Collocations

ABSTRACT: Deep learning has recently shown much promise for NLP applications. Traditionally, in most NLP approaches, documents or sentences are represented by a sparse bag-of-words representation. There is now a lot of work going beyond this by adopting a distributed representation of words, by constructing a so-called neural embedding or vector space representation of each word or document. This talk will go beyond the learning of word vectors. It will present methods for learning vector representations for Multiword Expressions and bilingual phrase pairs. All of these techniques, especially when integrated, are useful for various NLP applications.

SHORT BIOGRAPHY: Valia Kordoni joined the faculty of the Department of English and American Studies of the Humboldt-Universität zu Berlin (Germany) in November 2012. Her research interests include Computational Semantics, Deep Learning, and Machine Translation. She is the author of many refereed journal and conference publications and she has served as guest editor of many scientific journals. She was the Local Chair of ACL 2016 — The 54th Annual Meeting of the Association for Computational Linguistics. She has been the coordinator of many national-, EU- and industry-funded projects in Language Technology. Currently, she is the coordinator of "TraMOOC: Translation for Massive Open Online Courses" (`http://tramooc.eu`), which is a Horizon 2020 collaborative project aiming at providing reliable Machine Translation for Massive Open Online Courses (MOOCs). She has taught many invited tutorials, e.g., on"Robust Automated Natural Language Processing with Multiword Expressions and Collocations" in ACL 2013, on "Robust Semantic Analysis of Multiword Expressions with FrameNet" in EMNLP 2015, and very recently on "Deep Learning for Multiword Expressions and Collocations" in ACL 2017. Finally, she is also the author of "Multiword Expressions - From Linguistic Analysis to Language Technology Applications" (to appear in Springer).

Nikola Kompa University of Osnabrück, Germany

Invited Speaker

Language Evolution: Gricean Intentions meet Lewisian Conventions

ABSTRACT: How might language have evolved? The question will be addressed by first exploring differences between human language and animal communication systems; the difference between natural sigs, signals and symbols will be elaborated on. I will then claim that in order to explain how symbolic language emerged, it might prove helpful to combine Paul Grice's idea of non-natural meaning with David Lewis' game-theoretic model of conventions. More specifically, I will argue that a certain level of cooperation is needed if non-natural signs are to be interpretable at all; that iconicity helps with the problem of equilibrium selection; and that the idea of signaling problems as a type of coordination problem can explain how signs acquire specific yet stable meanings.

SHORT BIOGRAPHY: Nikola Kompa is a Professor of Theoretical Philosophy at the Institute of Philosophy, University of Osnabrück/Germany. She works on a variety of topics within philosophy of language and epistemology. She is particularly interested in theories of language comprehension, the relation

between language and cognition, metaphor and vagueness on the one hand and epistemic contextualism and relativism on the other.

Torbjörn Lager University of Gothenburg, Sweden

Invited Speaker

Make Prolog Great Again! Web logic programming for computational linguistics

ABSTRACT: A couple of decades ago, the Prolog programming language was quite popular in computational linguistics circles, but this is no longer the case. The evolution of Prolog has not stopped however, and has taken a direction that promises to once again make it highly relevant for computational linguistics. In this talk, I will present an extension of Prolog, called *Web Prolog*, that offers new types of concurrency and distribution, a datatype suitable for the representation of feature structures, new primitives suitable for dialogue management, "logic servers" that can be called from other programming languages, and an easy way to build web applications served by back-ends written in Web Prolog.

SHORT BIOGRAPHY: Torbjörn Lager is a professor of general and computational linguistics in the Department of Philosophy, Linguistics and Theory of Science at the University of Gothenburg. He has a broad range of interests within the field of computational linguistics, and previous work include the development of a logical approach to computational corpus linguistics, logic-based tools for machine learning, and platforms for the development of multimodal dialogue systems. In recent years he has found himself becoming more and more involved in the design and implementation of programming languages that offer high-level support in the intersection between the areas of language technology and web technology.

Staffan Larsson University of Gothenburg, Sweden

Invited Speaker

Modeling Intensions as Classifiers

ABSTRACT: Knowing the perceptual meaning of an expression allows an agent to identify perceived objects and situations falling under the meaning of the expression. We presently work towards a formal semantics for perceptual aspects of meaning, connecting these to logical-inferential aspects of meaning. The key ideas are (1) to model perceptual meanings as classifiers of perceptual input, (2) to regard classifiers in formal semantics as (parts of) the intensions of linguistic expressions, and (3) that speakers interactively train classifiers to arrive at shared meanings. We cast these ideas in Type Theory with Records. (TTR) and sketch how they relate to neural activity in (semi-)transparent neural networks.

SHORT BIOGRAPHY: I am Professor of Computational Linguistics at the Department of Philosophy, Linguistics and Theory of Science at the University of Gothenburg. I am also co-director of the Dialogue Technology Lab at the Centre for Language Technology (CLT), as well as CTO and co-founder and Chief Science Officer of Talkamatic AB. I am a member of the Editorial Board of the Journal Dialogue and Discourse. My areas of interest include dialogue, dialogue systems, language and cognition, pragmatics, formal semantics, semantic coordination, in-vehicle dialogue systems, philosophy of language.

Gleb Lobanov[1] **and Krasimir Angelov**[2]

[1]Immanuel Kant Baltic Federal University, [2]Chalmers University of Technology

and University of Gothenburg

Planning for Natural Language Generation in GF

ABSTRACT: In this paper, we present a minimalistic approach to document planning in natural language generation which is specifically targeted to the Grammatical Framework (GF) system, but it also generalizes to other formalisms with a type theoretical abstract syntax. The advantage of type theory is that it already has in place enough mechanisms to constrain the set of possible alternatives for planning the document. It turns out that document planning is easy to describe as a type theoretical proof search under a linear context. We present the method and demonstrate it within the use case of weather report generation.

Roussanka Loukanova Stockholm University, Sweden

Invited Speaker

Type Theory of Situated Algorithms and Information

ABSTRACT: We develop mathematical structures, which are type-theoretical models of situations, algoritms, and entities that carry, process, and memorize information in memory cells. The structures are computational, by modelling algorithms that have capacities for computing and storing information. We call these structures Typed Models of Situated Algorithms and Information. Concurrently, we develop respective formal languages having denotational and algorithmic semantics in such situated structures. Collectively, we call the classes of such formal languages, along with their semantics in algorithmic, situated structures, Type-Theory of Situated Algorithms and Information (TTofSAI). Formal languages of TTofSAI, with associated calculi, are essential for applications in computational approaches to language phenomena, including in Computational Linguistics. Prominently, TTofSAI opens new directions to algorithmic syntax-semantics interfaces. The talk will present the development of TTofSAI and its existing and potential applications to Computational Linguistics.

SHORT BIOGRAPHY: Roussanka Loukanova has master degrees in mathematics, from Sofia University (Bulgaria), and in computer science, from Indiana University Bloomington (US), and Ph.D. degree in mathematics from Moscow State University (Russia). She has been teaching at Sofia University (Bulgaria), Indiana University Bloomington (US), University of Minnesota (US), Illinois Wesleyan University (US), and Uppsala University (Sweden). Her research is in the areas of Typed Theory of Situated Information, Type Theory of Algorithms, Computational Syntax, Computational Semantics, Computational Syntax-Semantics Interface, Generalized computational grammar that includes lexicon and syntax-semantics interface, Type-Theoretic Grammars, Constraint Based Lexicalized Grammar (CBLG). The focus of her research is on development of type theory of information, language, and algorithms, for modeling phenomena of partiality, underspecification, and context dependency. She is a researcher at Stockholm University, Sweden.

Louise McNally Universitat Pompeu Fabra, Spain

Invited Speaker

On the Relation between Descriptive Content and Reference and its Implications for Computational Modeling

ABSTRACT: Although many formal models of natural language adhere to the hypothesis that natural language is compositional and that constituent structure mirrors semantic composition, there are well-known challenges to this hy-

pothesis, such as the seeming non-compositionality of certain types of idiomatic expressions (e.g. 'pull some/many/no strings for someone'). In this talk, based on joint work with Berit Gehrke, I argue that some of these problematic data support the development of language models in which the composition of complex descriptive contents should be able to proceed (though need not proceed) unmediated by reference. I further argue that the separation of descriptive content and reference on linguistically principled grounds should serve as a guide in developing equally principled combinations of statistical and symbolic approaches to computational modeling of language.

SHORT BIOGRAPHY: Louise McNally has Ph.D. University of California, Santa Cruz, 1992. She is Professor of Linguistics at Universitat Pompeu Fabra, Barcelona, Spain. Her research primarily concerns the interaction of lexical and compositional semantics, with particular focus on the syntax and semantics of modification. In recent years, she has collaborated with computational linguists in exploring distributional models of nominal phrases. She is currently working on integrating symbolic and statistical approaches to language modeling in order to develop a better understanding of how reference and conceptual knowledge interact in the construction of meaning.

Richard Moot CNRS, France

Invited Speaker

First-order Linear Logic for Natural Language Analysis

ABSTRACT: First-order linear logic is a proof-theoretically simple system which has many interesting applications in computational linguistics. I will show that first-order linear logic provides a very general 'machine language' underlying other formalisms, such as the Lambek calculus and lambda grammars. Using first-order linear logic as an underlying framework allows us to compare the analyses of different grammatical systems. The current implementation provides a theorem prover for all these grammatical systems, allowing grammar writers to specify grammars and to see the resulting analyses in their own framework, yet using first-order linear logic underneath as a simple and effective computational device.

SHORT BIOGRAPHY: Richard Moot has a Ph.D. in computational linguistics for Utrecht University. He is currently a CNRS research scientist at LIRMM in Montpellier studying type-logical grammars, proof theory, and the syntax-semantics interface. He has written the Grail family of theorem provers for type-logical grammars and is the author, with Christian Retoré, of the textbook *The Logic of Categorial Grammars*.

Glyn Morrill Universitat Politécnica de Catalunya, Spain

Invited Speaker

Parsing Logical Grammar: CatLog3

ABSTRACT: CatLog3 is a Prolog parser/theorem-prover for (type) logical (categorial) grammar. In such logical grammar, grammar is *reduced* to logic: a string of words is grammatical iff an associated logical statement is a theorem. CalLog3 implements a logic extending displacement calculus, a sublinear fragment including as primitive connectives the continuous (Lambek) and discontinuous wrapping connectives of the displacement calculus, additives, 1st order quantifiers, normal modalities, bracket modalities and subexponentials. In this paper we survey how CatLog3 is implemented on the principles of Andreoli's focusing and a generalisation of van Benthem's count-invariance.

Short Biography: Morrill received my degree, in Computer Science, from Cambridge in 1984 and my masters and doctorate, in Cognitive Science, from Edinburgh in 1985 and 1988. During a two-year postdoc in Edinburgh, he was struck by the notion of grammar as logic, which he has pursued ever since. Following a visiting scientist position at the University of Utrecht and CWI Amsterdam 1990–91, he went to the UPC, Barcelona, where he have remained. Morrill's books are *Type Logical Grammar: Categorial logic of signs* (Kluwer Academic Press, 1994), *Logica de Primer Ordre* (Edicions UPC, 2001), and *Categorial Grammar: Logical syntax, semantics and processing* (Oxford, 2011). Morrill has been awarded an ICREA Academia 2012 by the Catalan Institute for Research and Advanced Studies.

Joakim Nivre Uppsala University, Sweden

Invited Speaker

The Logic of Universal Dependencies

Abstract: Universal Dependencies (UD) is a framework for cross-linguistically consistent treebank annotation that has so far been applied to over 50 languages. A basic design principle of UD is to give priority to grammatical relations between content words, which are more likely to be parallel across languages, and to treat function words essentially as features of content words, functionally similar to but structurally distinct from morphological inflection. This principle has been questioned on the grounds that it gives rise to representations that are suboptimal for dependency parsing, where higher accuracy has often been observed when function words are treated as syntactic heads. In this talk, I will defend this principle from three different perspectives. First, I will show how it allows us to capture linguistic universals, similarities in grammatical constructions across structurally different languages, and thereby gives us a solid basis for contrastive linguistic studies. Second, I will illustrate how it provides a natural interface to semantic interpretation, and thereby serves the needs of downstream language understanding tasks, especially in multilingual settings. Finally, I will review recent work on UD parsing, suggesting that the suboptimal nature of the representations has been greatly exaggerated.

Short Biography: Joakim Nivre is Professor of Computational Linguistics at Uppsala University. He holds a Ph.D. in General Linguistics from the University of Gothenburg and a Ph.D. in Computer Science from Växjö University. His research focuses on data-driven methods for natural language processing, in particular for syntactic and semantic analysis. He is one of the main developers of the transition-based approach to syntactic dependency parsing, described in his 2006 book Inductive Dependency Parsing and implemented in the widely used MaltParser system, and one of the founders of the Universal Dependencies project, which aims to develop cross-linguistically consistent treebank annotation for many languages and currently involves over 100 researchers around the world. He has produced over 200 scientific publications and has more than 10,000 citations according to Google Scholar (May 2017). He is currently the president of the Association for Computational Linguistics.

Rainer Osswald Heinrich-Heine-Universität Düsseldorf, Germany

Invited Speaker

Frame-Semantic Composition at the Syntax-Semantics Interface

Abstract: Frames, understood as labelled attribute-value structures with types and relations, have recently gained new attention as representational

structures in linguistic semantics. The topic of the talk is the use of frames for the modelling of verb semantics and semantic composition in a number of verb-based constructions. The proposed model of the syntax-semantics interface combines semantic frames with syntactic trees, leaning on the formalism of Lexicalized Tree Adjoining Grammars. In this model, semantic composition is guided by syntactic operations and is basically realized via frame unification under constraints. A special focus of the talk will be on the logical description of frame representations.

SHORT BIOGRAPHY: Rainer Osswald is a senior researcher at the Collaborative Research Center *The Structure of Representations in Language, Cognition, and Science* located at Heinrich Heine University Düsseldorf, Germany. He received a Ph.D. in Computer Science from the University of Hagen (2002). The title of his thesis was "A Logic of Classification – with Applications to Linguistic Theory". Rainer Osswald is a member of the DIN/ISO standards committee on language resource management and he is currently the speaker of the Special Interest Group on Computational Linguistics of the German Linguistic Society. The focus of his current research is on the formal modelling of the syntax-semantics interface of verb-based constructions by means of decompositional frame semantics, Role and Reference Grammar, and Tree Adjoining Grammars.

Peter Pagin Stockholm University, Sweden

Invited Speaker

Switcher Semantics and Belief Sentences

ABSTRACT: Switcher semantics is a framework characterized by allowing that the semantic function that applies to an occurrence of a complex expression e is switched to another function as we move to a subexpression e' of e. A historical example is Frege's idea of indirect reference in indirect contexts. When such a function switch takes place, the semantics is no longer standard compositional, but it can still be compositional in a more general sense, which will be explained. A switcher semantics for belief sentences will also be presented, where structured meanings are the values relevant in belief contexts.

SHORT BIOGRAPHY: Peter Pagin is professor of theoretical philosophy at Stockholm University. He got his Ph.D. in 1987 with Dag Prawitz as supervisor. He works mainly in the philosophy of language and formal semantics. He has worked on compositionality, non-extensional contexts, vagueness, assertion, holism, among other topics. His most frequent co-authors are Dag Westerståhl and Kathrin Glüer.

Gerald Penn University of Toronto, Canada

Invited Speaker

What Does it Mean to Parse with Categorial Grammar?

ABSTRACT: There are many kinds of categorial grammar, but in practice there is very little variation in how categorial grammars are being used in practice by computational linguists. This talk will reassess the several arguments that have been offered in defense of the status quo in light of research on CG membership algorithms and corpora over the last 10 years.

Michael Richter and Roeland van Hout Radboud University, Nijmegen

How WIE 'how' as Intensifier Co-occurs with other Intensifiers in German Sentence

ABSTRACT: In this paper we state that (1) in exclamative sentences such as 'wie cool war das denn!' 'how cool was that', wie (degree related adverbial wh-exclamatives, Nouwen and Chernilovskaya 2015) is an intensifier of the gradable adjective cool (see Rett 2015, on degrees within exclamatives) and that (2) wie 'how' can co-occur with (gradable adjectival) intensifiers of similar semantic properties, while it cannot with intensifiers with different properties.

Maria Skeppstedt[1], **Kostiantyn Kucher**[1], **Carita Paradis**[2], **Andreas Kerren**[1]
[1]Linnaeus University, Växjö, [2]Lund University, Sweden

Language Processing Components of the StaViCTA Project

ABSTRACT: The StaViCTA project is concerned with visualising the expression of stance in written text, and is therefore dependent on components for stance detection. These components are to (i) download and extract text from any HTML page and segment it into sentences, (ii) classify each sentence with respect to twelve different, notionally motivated, stance categories, and (iii) provide a RESTful HTTP API for communication with the visualisation components. The stance categories are certainty, uncertainty, contrast, recommendation, volition, prediction, agreement, disagreement, tact, rudeness, hypotheticality, and source of knowledge.

Sam Sanders Ludwig-Maximilians University of Munich, Germany

Invited Speaker

A Computable Solution to Partee's Temperature Puzzle (joint work with Kristina Liefke)

ABSTRACT: We present a computable solution to Partee's temperature puzzle which uses one of the standard tools of mathematics and the exact sciences: countable approximation. Our solution improves upon the standard Montagovian solution to the puzzle (i) by providing computable natural language interpretations for this solution, (ii) by lowering the complexity of the types in the puzzle's interpretation, and (iii) by acknowledging the role of linguistic and communicative context in this interpretation. These improvements are made possible by interpreting natural language in a model that is inspired by the Kleene-Kreisel model of countable-continuous functionals. In this model, continuous functionals are represented by lower-type objects, called the *associates* of these functionals, which only contain countable information.

SHORT BIOGRAPHY: Sam Sanders obtained a Ph.D. in mathematics at Ghent University (Belgium) in May 2010. His current research interests are centered around computability, broadly construed, and are at the intersection of logic, philosophy, linguistics, and computer science.

Mila Vulchanova Norwegian University of Science and Technology, Norway

Invited Speaker

Figurative Language Processing: a bottleneck in developmental deficits and NLP

ABSTRACT: It is now common to employ evidence from human behaviour (e.g., child development) for the creation of computational models of this behaviour with a variety of applications (e.g., in developmental robotics). This talk will report comprehensive research in the comprehension and processing of figurative (non-literal) language in highly verbal individuals with autism in comparison with age- and language level-matched neuro-typical individuals. Based on

this evidence we will outline the strategies used by human language users in understanding non-literal/non-compositional expressions and proceed to identify possible solutions for automated language systems in the domain of idiomatic expressions.

Short Biography: Prof. Mila Vulchanova is the Director of the Language Acquisition and Language Processing Lab and the Scientific Director of the Norwegian Graduate School of Linguistics and Philology. She has expertise in theoretical linguistics, language and cognition, language learning and language comprehension across the life-span, and language and cognition in developmental deficits. She has published in Psychological Science, PLoS ONE, Cognitive Neuropsychology, Learning and Individual Differences, Frontiers in Human Neurosceince, and has edited a number of volumes with Oxford University Press, John Benjamins and as guest editor for journals. She has both coordinated and participates in big-scale EU projects, such as Language & Perception (LanPercept), Decitic Communication (DCOMM) and a number of COST Actions. She is an elected member of the Norwegian Royal Academy for Science and Letters (DKNVS) since 2002.

Markus Werning Ruhr University Bochum, Germany

Invited Speaker

Compositionality in a World Without Synonyms — Existence and Uniqueness of Similarity-based Semantics

Abstract: Compositionality is the principle that the meaning of a complex expression is determined by the meanings of its syntactic parts and the way these parts are combined. Under a standard formalization, it is equivalent to the proposition that, if there are pairs of synonyms in a language, the paired expressions are simultaneously interchangeable in any complex meaningful expression without changing the meaning of that expression. It has been argued on various grounds, though, that there are strictly speaking no two expressions (at least in languages like English or German) that fulfill this condition. To avoid the vacuity of the principle, it would hence make sense to formulate an analogous principle that is not based on strict synonymy, but on meaning similarity. Leitgeb (2008), however, has proved that, under reasonable assumptions, such a notion of meaning similarity is not tenable for a bi-valued propositional system. In the paper to be presented, we lift the requirement of bi-valuedness and proof an existence and uniqueness theorem for a similarity-based compositional semantics in a many-valued propositional system.

Short Biography: Markus Werning's agenda as a philosopher is deeply rooted in a naturalistic understanding of philosophy. The overarching goal of his research is to approach questions in the philosophy of language and mind and related areas (epistemology, descriptive metaphysics, etc.) (i) with exact theoretical methods – e.g. logic, formal semantics, and probability theory –, (ii) with advanced empirical methods such as EEG and fMRI, and (iii) by computational modelling. He views those areas of philosophy in continuity with the cognitive sciences where, in philosophy, emphasis is put on theoretical rigor as well as meta- and cross-disciplinary aspects. His naturalistic attitude towards philosophy arises from a general skepticism with regard to apriori reasoning such as conceptual analysis, intuition, and introspection as a privileged source of knowledge.

Gijs Wijnholds and Mehrnoosh Sadrzadeh Queen Mary University of London

Vector Semantics for Lambek Calculus with Contraction

ABSTRACT: (For the references, see the paper in the Proceedings of LA-CompLing2017)

There are linguistic phenomena that, next to the movement of constituents, involve a specific form of copying and/or deletion of information, as argued for in [2, 6, 7]. Notable examples are pronoun relativisation [5, 6] and para- sitic gaps [6], iterated coordination [7] and ellipsis [2]. Whether the burden of such phenomena should lie on the lexicon or on the syntactic process depends on the particularities of the phenomenon in question and the language in which it manifests itself. For instance, there are cases of pronoun relativisation that exemplify a form of derivational ambiguity ('mannen die vrouwen haten' in Dutch) whereas the English equivalent does not occur as an ambiguity but rather comes with a long distance dependency dealt with by different lexical instances (viz. compare 'men who hate women' and 'men whom women hate'). The case of (verb) ellipsis traditionally has been approached both as a syntactic problem as well as a semantic one [4]. Recent work in distributional semantics has shown that Frobenius algebras over vector spaces can deal with ellipsis [3] and relative pronouns [1]. However, they are introduced as meaning postulates which means that they can arise only through lexical semantics and not as part of the syntactic process. Similar to the approaches of [2, 6, 7], we argue for the use of controlled forms of copying in a type logical system, so that we regain the balance between derivational and lexical ambiguity, in the presence of these complex semantic operations. In order to facilitate a controlled form of copying, we define an enrichment of the Lambek Calculus with control modalities that facilitates contraction on modally decorated formulas. The presence of such a structural rule in an uncontrolled/global version would obliterate the distinguishing power of the original calculus, whereas the use of modalities gives explicit control over when these operators are licensed. We show how the Frobenius algebras used in previous work [1, 3] provide vector semantic counterparts for the proof theoretic copying rule of our system. Moreover, we show how our system related to the systems of [2, 6, 7]. The vector space semantics is given in the style of a Curry-Howard annotation: words are interpreted as vectors, proofs as linear maps between vector spaces. Semantic content is obtained from big data (not even limited to textual data) on top of which we describe the meaning of phrases beyond words by compositional interpretation from syntax into semantics. We illustrate with vector computations for ellipsis cases like 'Mary writes Prolog and Gary does too' and 'Gary loves his car and Bill too'.

Yukiko Yana[1], Koji Mineshima[1,2] and Daisuke Bekki[1,2]
[1]Ochanomizu University, Japan; [2]CREST, Japan Science and Technology Agency, Japan

Variable Handling in DRT and DTS

ABSTRACT: This paper discusses the differences in the behaviours of Discourse Representation Theory (DRT) and Dependent Type Semantics (DTS) with respect to variable handling. Since substitution in DRT is only partially defined, so is β-reduction and α-conversion, which brings about the overwrite problem (it is also called destructive update) and the duplication problem. We also compare the status of DRT with that of DTS and show that these two problems do not arise in DTS.

Part II

Contributions of Papers

Phase Spaces for Involutive Nonassociative Lambek Calculus

Wojciech Buszkowski
The Adam Mickiewicz University in Poznań

Abstract

Involutive Nonassociative Lambek Calculus is a nonassociative version of Noncommutative MLL [1], but the multiplicative constants are not admitted. It extends Nonassociative Lambek Calculus (NL) by two linear negations. InNL is a strongly conservative extension of NL. We define and study phase spaces for InNL. Using them, we prove the cut-elimination theorem for a one-sided sequent system for InNL; in fact, we prove the completeness theorem for a cut-free system. In a similar way, we prove completeness and cut elimination for an auxiliary system InNL(k), where $k \geq 2$ is even, an analogue of CNL from [22, 12]; CNL amounts to InNL(2). These proofs also work for richer systems with additives. Using InNL(k), we show that InNL is P-TIME (another proof was given in [13]) and the type grammars based on InNL generate the ($\epsilon-$free) context-free languages. These results remain true for the version of InNL with multiplicative constants.

1 Introduction and preliminaries

Type grammars (categorial grammars) are formal grammars which describe a language by assigning types to lexical atoms (words). Parsing is based on a fixed type logic, independent of the particular language. This embodies the paradigm of *parsing as deduction*. Types are formulas of the given type logic. Natural deduction systems for type logics strictly correspond to lambda terms in different versions of the lambda calculus (via the Curry-Howard isomorphism), which naturally connects proof-theoretic semantics with referential semantics; see van Benthem [7], Moortgat [28], Morrill [30].

The so-called Basic Categorial Grammars (BCGs), originated by Ajdukiewicz [3] and Bar-Hillel [4, 5], employ a simple redution procedure, based on the application laws $A, A\backslash B \Rightarrow B$ and $A/B, B \Rightarrow A$. This purely applicative logic was extended by Lambek [24] to Syntactic Calculus, admitting the connectives $\otimes$ (product), $\backslash$ (right division), and $/$ (left division), which satisfy the residuation rules: $A \otimes B \Rightarrow C$ iff $B \Rightarrow A\backslash C$ iff $A \Rightarrow C/B$, and the associative law for $\otimes$. The application laws take the forms $A \otimes (A\backslash B) \Rightarrow B$ and $(A/B) \otimes B \Rightarrow A$. One obtains new laws, e.g. $A \Rightarrow (B/A)\backslash B$, $A \Rightarrow B/(A\backslash B)$

(type raising), $A \Rightarrow B\backslash(B \otimes A)$, $A \Rightarrow (A \otimes B)/B$ (co-application), and $(A\backslash B) \otimes (B\backslash C) \Rightarrow A\backslash C$, $(A/B) \otimes (B/C) \Rightarrow A/C$ (composition).

Standard models for Syntactic Calculus are algebras of $\epsilon-$free languages on an alphabet (lexicon) Σ; $\Rightarrow$ is interpreted as set inclusion and $\otimes, \backslash, /$ as follows.

$$L_1 \cdot L_2 = \{uv : u \in L_1, v \in L_2\}$$

$$L_1\backslash L_2 = \{v \in \Sigma^+ : L_1 \cdot \{v\} \subseteq L_2\},\ L_1/L_2 = \{u \in \Sigma^+ : \{u\} \cdot L_2 \subseteq L_1\}$$

For example, if s, pn, and n are the categories of sentence, proper noun, and common noun, respectively (understood as some sets of strings), then $pn\backslash s$ can be interpreted as the category of verb phrase, $(pn\backslash s)/pn$ - transitive verb phrase, $np = s/(pn\backslash s)$ - noun phrase, and np/n - determiners. So the expression 'John admires Jane' can be assigned s, since the sequent

$$pn, (pn\backslash s)/pn, pn \Rightarrow s$$

is provable in L (in fact, in the purely applicative logic), whereas 'he admires her' leads to the sequent

$$s/(pn\backslash s), (pn\backslash s)/pn, (s/pn)\backslash s \Rightarrow s$$

provable in L with the aid of the composition laws.

These plane examples ignore fine aspects of grammar (tense, number, agreement etc.). We refer the reader to [28, 30, 31] for a deeper linguistic material and other interpretations, e.g. type-theoretic semantics.

Nowadays Syntactic Calculus is called Lambek Calculus (L) and its nonassociative version, due to Lambek [25], Nonassociative Lambek Calculus (NL). (The latter does not support the composition laws.) Both L and NL are treated as basic type logics for type grammars; L parses expressions represented as strings, whereas NL corresponds to phrase structures. One also considers different extensions of these logics, e.g. with multiplicative constants 1, 0, additives $\wedge, \vee, \top, \bot$, unary modal operators $\Diamond$, $\Box$, their residuals $\Box^{\downarrow}$, $\Diamond^{\downarrow}$, and others.

Product (also called multiplicative conjunction) need not satisfy $A \Rightarrow A\otimes A$ (contraction), $A\otimes B \Rightarrow A$, $A\otimes B \Rightarrow B$ (weakening), nor $A\otimes B \Rightarrow B\otimes A$ (exchange). These laws correspond to three structural rules, admitted in standard sequent systems for classical and intuitionistic logics. The logics that omit at least some of these rules are called *substructural logics* [19]. Some authors define substructural logics as axiomatic and rule extensions of L with 1, $\wedge$, $\vee$ (Full Lambek Calculus), but weaker systems may also be counted to this family. It contains many well-known non-classical logics, e.g. many-valued logics, fuzzy logics, relevance logics, with intuitionistic and classical logics in the limit.

With exchange, one proves $A\backslash B \Leftrightarrow B/A$ (i.e. both $\Rightarrow$ and $\Leftarrow$) and writes $A \rightarrow B$ for $A\backslash B$. Thus, implication $\rightarrow$ is a residual operation for

the commutative product. If $\otimes$ is not commutative, then $\rightarrow$ splits in two implications $\backslash$ and $/$, also written as $\rightarrow$ and $\leftarrow$; hence we call them the right and the left implication, or division, respectively.

With constant 0, one defines negation $\neg A = A \rightarrow 0$ or, for the non-commutative case, the right negation $\sim A = A\backslash 0$ and the left negation $-A = 0/A$. Assuming the double negation law $\neg\neg A \Rightarrow A$ or $\sim -A \Rightarrow A$, $- \sim A \Rightarrow A$ (the converse laws are provable), one obtains the so-called 'classical' versions of these logics. This term is misleading, since they remain non-classical in general, i.e. essentially weaker than classical logic. Therefore in this paper we use the term 'involutive', after [19, 18].

The term 'substructural logic' appeared in the literature in the early 1990-ties. Earlier, Girard [20] introduced linear logics: Propositional Linear Logic is equivalent to Full Lambek Calculus with exchange, 'classical' negation and exponentials !, ? (unary modalities allowing the execution of structural rules for modal formulas). The fragment without exponentials is called Multiplicative-Additive Linear Logic (MALL) and MALL without $\wedge, \vee$ Multiplicative Linear Logic (MLL). Noncommutative MALL with one negation $\neg A = A\backslash 0 = 0/A$ was studied by Yetter [35] and with two negations by Abrusci [1]. The latter was called (Classical) Bilinear Logic by J. Lambek; see [26]. The former is also referred to as Cyclic Noncommutative MALL [2].

The present paper does not use Girard's original notation, which is non-orthodox (though widely adopted in the linear logic community). For example, Girard writes & for our $\wedge$ but $\oplus$ for our $\vee$, $\top$ for the top element but 0 for the bottom element (our $\bot$), 1 for the multiplicative unit (the unit element for $\otimes$) but $\bot$ for its dual (our 0, i.e. the unit element for the dual product). The notation used here is standard in the literature on substructural logics except that in what follows we write $A^{\sim}$ for $\sim A$ and A^{-} for $-A$, just to shorten formulas (linear logicians write $A^{\bot}$ and ${}^{\bot}A$). In particular, $\oplus$ denotes here dual product (par), defined by: $A \oplus B = (B^{\sim} \otimes A^{\sim})^{-}$.

From the very beginning, linear logics were regarded as logics closely related to type logics of type grammars. MALL is a conservative extension of (Full) Lambek Calculus with exchange, and both noncommutative versions of MALL are conservative extensions of (Full) Lambek Calculus [2]. Often Lambek calculi are characterized as the intuitionistic fragments of the corresponding linear logics.

Due to conservativity, every type grammar based on L with 1 can be translated into a grammar based on a linear logic, and the latter has precisely the same expressive and generative power as the former. Some authors directly apply noncommutative linear logics in type grammars; see e.g. [14, 15]. Pregroup grammars, due to Lambek [27], are based on Compact Bilinear Logic, i.e. an extension of Bilinear Logic, where $\otimes$ and $\oplus$ (hence also 1 and 0) collapse. Proof nets (a graph-theoretic representation of proofs in MLL) were adapted for noncommutative linear logics and Lambek calculi

and used as logical forms of expressions [31]. The role of exponentials in PLL stimulated the study of multi-modal extensions of L and NL, where different modal operators (modes) are introduced for a controlled usage of structural rules (associativity, exchange); see [28, 31].

All linear logics, mentioned above, admit the associative law for $\otimes$. Nonassociative linear logics were not studied so extensively. de Groote and Lamarche [22] introduced Classical Nonassociative Lambek Calculus (CNL), being a nonassociative version of Cyclic Noncommutative MLL. CNL does not admit multiplicative constants. Their work is purely proof-theoretic, focusing on proof nets for CNL. They, however, present a one-sided sequent system for CNL, prove the cut-elimination theorem (using proof nets) and the P-TIME complexity of CNL; they also show that CNL is a conservative extension of NL.

[12] continues this research. It defines phase spaces for CNL and employs them to prove cut elimination for a sequent system (dual to that of [22]) and the strong finite model property for CNL. It strengthens some results of [22]: (1) CNL is a strongly conservative extension of NL, (2) the finitary consequence relation for CNL is P-TIME. [12] also shows (3): the type grammars based on CNL generate the $\epsilon-$free context-free languages. The proof of (1) employs a simple construction of a phase space and can easily be adapted for richer logics (with additives, multiplicative constants etc.). In particular, one can prove in this way that both versions of noncommutative MALL are strongly conservative extensions of Full Lambek Calculus; [2] shows the weak conservativity by more involved proof-theoretic tools.

Phase spaces are certain frames (like Kripke frames for modal logics), used as basic models for linear logics. For nonassociative logics without multiplicative constants they can be defined more generally, which makes their relation to modal frames more explicit; see Section 2.

Recall that a logic possesses the finite model property, if it is complete with respect to its finite models, and the strong finite model property, if it is strongly complete with respect to its finite models. Models can be understood as abstract algebras (see below). The (resp. strong) completeness means that the formulas (or sequents) provable (resp. from a fixed set of assumptions) in this logic are precisely those which are valid in all models of this logic (resp. true in all models for any valuation which satisfies the assumptions). Let $\mathcal{L}$ and $\mathcal{L}'$ be logics such that the language of $\mathcal{L}'$ be richer than that of $\mathcal{L}$. $\mathcal{L}'$ is called a conservative extension of $\mathcal{L}$, if any formula (or sequent) in the language of $\mathcal{L}$ is provable in $\mathcal{L}$ if and only if it is provable in $\mathcal{L}'$, and a strongly conservative extension of $\mathcal{L}$, if this also holds for the provability from assumptions (in the language of $\mathcal{L}$). The finitary consequence relation means the provability from a finite set of assumptions.

The completeness shows that our logic adequately describes the intended reality and, for logics considered here, the (resp. strong) finite model property implies the decidability of (resp. the finitary consequence relation for)

the given logic. If we know that the logic is decidable, then we may estimate its complexity, and we go this way in [12] and here.

In the present paper we study a weaker logic InNL, i.e. NL augmented with two negations $^{\sim}$,$^{-}$, satisfying the following laws.

$$A^{\sim -} \Leftrightarrow A\,,\ A^{-\sim} \Leftrightarrow A \text{ (the double negation laws)} \tag{1}$$

$$A^{\sim}/B \Leftrightarrow A\backslash B^{-} \text{ (the contraposition law)} \tag{2}$$

CNL amounts to InNL with $A^{\sim} = A^{-}$. InNL is a nonassociative version of Noncommutative MLL [1]. CNL and InNL do not admit the multiplicative constants; the corresponding algebras need not have unit elements for product and dual product. Their extensions with these constants are denoted here by CNL1 and InNL1. Following the terminology of [19], CNL may also be called Cyclic InNL (CyInNL).

Although CNL and InNL are similar in many aspects, there is an essential difference: CNL possesses the strong finite model property, but InNL does not [12]. For $A^{\sim} \Rightarrow A^{-}$ entails $A^{-} \Rightarrow A^{\sim}$ in finite models (for any fixed A) but not in arbitrary models. Accordingly, not all methods applicable to CNL can be adapted for InNL. In particular, one can form finite sets of formulas closed under the CNL-negation (up to logical equivalence), which is impossible for the InNL-negations. We prove here that the pure InNL is P-TIME, but the complexity of the (finitary) consequence relation for InNL is left as an open problem.

InNL1 with additive (extensional) connectives $\wedge, \vee$ amounts to InGL (involutive groupoid logic) from [18]; it is a conservative extension of InNL1. [18] provides a two-sided sequent system for InGL, admitting cut elimination. By the subformula property, this system restricted to multiplicative connectives and constants is good for InNL1. We, however, present a simpler, one-sided sequent system, analogous to the system in [1] for Noncommutative MLL. This system seems more convenient for proof search and for proving metalogical theorems (decidability, finite model property).

Like in [12] for CNL, we define phase spaces for InNL and apply them in the proofs of our results, e.g. the cut-elimination theorem. They are also used to prove that the provability in InNL can be reduced to that in an auxiliary system InNL(k), whose (finitary) consequence relation is P-TIME; the type grammars based on InNL(k) generate the context-free languages. This yields the P-TIME complexity of InNL and the equivalence with context-free grammars.

The forthcoming paper [13] studies InNL by purely proof-theoretic tools. It contains a syntactic proof of the cut-elimination theorem and another proof of P-TIME comnplexity, not involving InNL(k). Phase spaces are not considered at all. The present paper fills up this gap.

Analogous results were earlier obtained for NL and its extensions (with additives, classical connectives, intuitionistic connectives, multiplicative constants, modal operators), using similar methods; see [9, 10]. Logics with

linear negation(s), however, require some refinements, elaborated in [12, 13] and here. Let us note that NL with $\wedge, \vee$ is coNP-hard [11], and the finitary consequence relation for this logic is undecidable [16]; it remains decidable, if the distributive laws for $\wedge, \vee$ are assumed [10].

Now, we briefly describe the algebraic models of our logics.

The algebraic models of NL are *residuated groupoids*, i.e. (ordered) algebras $\mathbf{M} = (M, \otimes, \backslash, /, \leq)$ such that $(M, \leq)$ is a poset, and $\otimes, \backslash, /$ are binary operations on M, satisfying *the residuation laws*:

$$a \otimes b \leq c \text{ iff } b \leq a\backslash c \text{ iff } a \leq c/b$$

for all $a, b, c \in M$. A residuated groupoid $\mathbf{M}$ is *unital*, if it contains an element 1 such that $1 \otimes a = a = a \otimes 1$, for any $a \in M$. For any residuated groupoid $\mathbf{M}$, the product $\otimes$ is isotone in both arguments, hence $(M, \otimes, \leq)$ is a p.o. groupoid. We refer to $\backslash, /$ as the residual operations for product.

The algebraic models of InNL are *involutive residuated groupoids*, i.e. algebras $\mathbf{M} = (M, \otimes, \backslash, /, ^{\sim}, ^{-}, \leq)$ such that $(M, \otimes, \backslash, /, \leq)$ is a residuated groupoid, and $^{\sim}$,$^{-}$ are antitone (i.e. order-reversing) unary operations on M, satisfying the double negation laws and the contraposition law:

$$\text{(DN) } a^{\sim -} = a = a^{- \sim},$$

$$\text{(CON) } a^{\sim}/b = a\backslash b^{-}$$

for all $a, b \in M$. One easily derives other contraposition laws: $a\backslash b = a^{\sim}/b^{\sim}$, $a/b = a^{-}\backslash b^{-}$. In a unital involutive residuated groupoid, there holds $1^{\sim} = 1^{-}$, since $1^{\sim} = 1^{\sim}/1 = 1\backslash 1^{-} = 1^{-}$ (in any unital residuated groupoid, $a/1 = a = 1\backslash a$). One defines $0 = 1^{\sim}$.

One shows: $(b^{-} \otimes a^{-})^{\sim} = (b^{\sim} \otimes a^{\sim})^{-}$, for all a, b, and defines the dual product: $a \oplus b = (b^{-} \otimes a^{-})^{\sim}$ (also called *par* in the literature on linear logics). If $1 \in M$, then $0 \oplus a = a = a \oplus 0$. One obtains: $a\backslash b = a^{\sim} \oplus b$, $a/b = a \oplus b^{-}$. Hence $a\backslash b = (b^{-} \otimes a)^{\sim}$, $a/b = (b \otimes a^{\sim})^{-}$. In unital involutive residuated groupoids, there hold: $a^{\sim} = a\backslash 0$, $a^{-} = 0/a$.

An involutive residuated groupoid is said to be *cyclic*, if $a^{\sim} = a^{-}$, for any element a. Cyclic involutive residuated groupoids are algebraic models of CNL; see [12].

A residuated semigroup is an associative residuated groupoid (i.e. $\otimes$ is associative), and *a residuated monoid* is a unital residuated semigroup. One also considers algebras with operations $\wedge, \vee$ (meet and join), satisfying the lattice laws. *A residuated lattice* is a lattice-ordered residuated monoid. Residuated lattices are the algebraic models of Full Lambek Calculus.

Now, we present NL and InNL as intuitionistic sequent systems. The systems with negation(s) do not admit cut elimination.

The formulas of NL are built from variables $p, q, r, \ldots$ by means of connectives $\otimes, \backslash, /$. One defines *bunches*: (i) every formula is a bunch, (ii) if Γ

and Δ are bunches, then (Γ, Δ) is a bunch. Bunches can be treated as the elements of the free groupoid generated by the set of formulas. *NL-sequents* are of the form $\Gamma \Rightarrow A$, where Γ is a bunch and A is a formula. *A context* is a bunch containing a special atom x (a place for substitution). We denote formulas by $A, B, C, \ldots$, bunches by Γ, Δ, Θ, and contexts by $\Gamma[\,], \Delta[\,]$ etc. $\Gamma[\Delta]$ denotes the result of substituting Δ for x in $\Gamma[\,]$. The axioms and the inference rules of NL are as follows.

$$\text{(NL-id) } A \Rightarrow A \quad \text{(NL-cut) } \frac{\Gamma[A] \Rightarrow B \;\; \Delta \Rightarrow A}{\Gamma[\Delta] \Rightarrow B}$$

$$(\otimes \Rightarrow) \; \frac{\Gamma[(A, B)] \Rightarrow C}{\Gamma[A \otimes B] \Rightarrow C} \quad (\Rightarrow \otimes) \; \frac{\Gamma \Rightarrow A \;\; \Delta \Rightarrow B}{(\Gamma, \Delta) \Rightarrow A \otimes B}$$

$$(\backslash \Rightarrow) \; \frac{\Gamma[B] \Rightarrow C \;\; \Delta \Rightarrow A}{\Gamma[(\Delta, A\backslash B)] \Rightarrow C} \quad (\Rightarrow \backslash) \; \frac{(A, \Gamma) \Rightarrow B}{\Gamma \Rightarrow A\backslash B}$$

$$(/ \Rightarrow) \; \frac{\Gamma[A] \Rightarrow C \;\; \Delta \Rightarrow B}{\Gamma[(A/B, \Delta)] \Rightarrow C} \quad (\Rightarrow /) \; \frac{(\Gamma, B) \Rightarrow A}{\Gamma \Rightarrow A/B}$$

This sequent system is due to Lambek [25]. NL1 is obtained by admitting the empty bunch ϵ, satisfying $(\epsilon, \Gamma) = \Gamma = (\Gamma, \epsilon)$, and the constant 1 (an atom) with two new rules and one new axiom.

$$(1 \Rightarrow) \; \frac{\Gamma[\Delta] \Rightarrow A}{\Gamma[(1, \Delta)] \Rightarrow A} \quad \frac{\Gamma[\Delta] \Rightarrow A}{\Gamma[(\Delta, 1)] \Rightarrow A} \quad (\Rightarrow 1) \; \epsilon \Rightarrow 1$$

We write $\Rightarrow A$ for $\epsilon \Rightarrow A$. NL1 is not a conservative extension of NL; $p/(q/q) \Rightarrow p$ is provable in NL1 but not in NL. Lambek [25] proved the cut-elimination theorem: every sequent provable in NL is provable without (NL-cut). His standard, syntactic proof can easily be adapted for NL1.

InNL (resp. InNL1) can be presented as an extension of NL (resp. NL1) with new unary connectives $^\sim, ^-$, new axioms (1), (2), and new inference rules:

$$\text{(r-CON) } \frac{A \Rightarrow B}{B^\sim \Rightarrow A^\sim} \quad \frac{A \Rightarrow B}{B^- \Rightarrow A^-}\,.$$

Notice that in InNL1 (r-CON) can be omitted; they are derivable in the resulting system. CNL (resp. CNL1) can be obtained from InNL (resp. InNL1) by adding the axioms $A^\sim \Leftrightarrow A^-$.

One easily shows that NL is strongly complete with respect to residuated groupoids, NL1 with respect to unital residuated groupoids, and InNL (resp. InNL1) with respect to (resp. unital) involutive residuated groupoids. Recall that *a valuation* in an algebra $\mathbf{M}$ is a homomorphism from the formula algebra to $\mathbf{M}$. It is extended for bunches by interpreting each comma as product. If the empty bunch is admitted, it is interpreted as 1. $\Gamma \Rightarrow A$ is *true* for a valuation μ in $\mathbf{M}$ if $\mu(\Gamma) \leq \mu(A)$.

[12] shows that CNL is *a strongly conservative extension* of NL. Also CNL1 is a strongly conservative extension of NL1. Since InNL (resp. InNL1) is intermediate between NL (resp. NL1) and CNL (resp. CNL1), then evidently InNL (resp. InNL1) is a strongly conservative extension of NL (resp. NL1).

NL is a basic logic of type grammars: formulas are interpreted as syntactic types of expressions, represented as phrase structures [28].

Now, we can precisely define type grammars. Let $\mathcal{L}$ be a logic, which admits sequents $\Gamma \Rightarrow A$. *A type grammar* (based on $\mathcal{L}$) is defined as a triple $G = (\Sigma, I, A_0)$ such that Σ is a finite alphabet (lexicon), I is a map which assigns a finite set of types (i.e. formulas of $\mathcal{L}$) to each $v \in \Sigma$, and A_0 is a designated type. One says that G *assigns* type A to the string $v_1 \dots v_n$, where $v_i \in \Sigma$, if there exist types $A_1, \dots, A_n$ such that $A_i \in I(v_i)$ for $i = 1, \dots, n$ and $A_1, \dots, A_n \Rightarrow A$ (for nonassociative logics, one adds: under some bracketing of $A_1, \dots, A_n$). *The language of* G, denoted by $L(G)$, consists of all $a \in \Sigma^+$ such that G assigns A_0 to a. Two classes of grammars are said to be (weakly) *equivalent*, if they yield the same classes of languages.

Several basic classes of type grammars are equivalent to the class of $\epsilon-$free context-free grammars, e.g. Basic Categorial Grammars, the type grammars based on NL, NL1, L, L1, NL with operations of a distributive lattice, CNL, pregroup grammars, and others (also InNL, InNL1, as it will be shown here). This, however, does not mean that type grammars can be identified with context-free grammars. Usually, the former are not strongly equivalent to the latter, e.g. they do not yield the same phrase structure languages. Furthermore, type grammars (except pregroup grammars) are closely related to type-theoretic semantics. Nonetheless, due to this equivalence, different efficient parsing methods for context-free grammars can be applied to type grammars.

Moortgat [29] considers an extension of NL with $\oplus$, its (dual) residuals and some mixed associativity and commutativity axioms, due to Grishin [21], under the name Lambek-Grishin Calculus (LG). The corresponding type grammars are called symmetric type grammars. [6] contains a detailed study of these grammars; one section discusses InNL, called there Nonassociative Bilinear Logic. In fact, all new axioms of LG are provable in the associative and commutative version of InNL. Let us note that in InNL with Grishin axioms:

$$A \otimes (B \oplus C) \Rightarrow (A \otimes B) \oplus C \text{ and } (A \oplus B) \otimes C \Rightarrow A \oplus (B \otimes C)$$

one proves the associative laws for $\otimes$ and $\oplus$, hence the resulting logic amounts to Noncommutative MLL without constants.

2 Phase spaces

By *a phase space* we mean a frame $(M, \cdot, R)$ such that $(M, \cdot)$ is a groupoid and $R \subseteq M^2$. For $X \subseteq M$ we define:

$$X^{\sim} = \{b \in M : \forall_{a \in X}\, R(a,b)\},\ X^{-} = \{a \in M : \forall_{b \in X}\, R(a,b)\}.$$

This yields a Galois connection: $X \subseteq Y^{\sim}$ iff $Y \subseteq X^{-}$, for all $X, Y \subseteq M$. Consequently, $^{\sim}$,$^{-}$ reverse $\subseteq$ and $X^{\sim -\sim} = X^{\sim}$, $X^{-\sim -} = X^{-}$, for any $X \subseteq M$. The operations $\phi_R(X) = X^{-\sim}$ and $\psi_R(X) = X^{\sim -}$ are closure operations on $\mathcal{P}(M)$. Recall that an operation C on $\mathcal{P}(M)$ is called *a closure operation*, if it satisfies: (C1) $X \subseteq C(X)$, (C2) if $X \subseteq Y$ then $C(X) \subseteq C(Y)$, (C3) $C(C(X)) = C(X)$, for all $X, Y \subseteq M$. A set $X \subseteq M$ is said to be $C-$*closed*, if $X = C(X)$. A set X is ϕ_R-closed (resp. ψ_R-closed) if and only if $X = Y^{\sim}$ (resp. $X = Y^{-}$) for some $Y \subseteq M$.

Given a groupoid $(M, \cdot)$ and $X, Y \subseteq M$, one defines:

$$X \cdot Y = \{a \cdot b : a \in X, b \in Y\},$$

$$X \backslash Y = \{b \in M : X \cdot \{b\} \subseteq Y\},\ X/Y = \{a \in M : \{a\} \cdot Y \subseteq X\}.$$

$\mathcal{P}(M)$ with these operations and $\subseteq$ is a residuated groupoid.

The symbols ϕ_R, ψ_R appeared in Abrusci [1] who studied phase spaces for Noncommutative MALL. Like for other linear logics, his phase space is of the form $(M, \cdot, O)$, where $O \subseteq M$ (linear logicians write $\perp^{\bullet}$ for our O). One can define $R_O = \{(a,b) \in M^2 : a \cdot b \in O\}$. Then $X^{\sim} = X \backslash O$, $X^{-} = O/X$. Our notion is more general and natural for logics without multiplicative constants. [12] provides an example of a commutative, associative phase space $(M, \cdot, R)$ for CNL (see below) such that $R \neq R_O$, for any $O \subseteq M$. However, if $(M, \cdot)$ is a free groupoid, then always $R = R_O$ for $O = \{a \cdot b : R(a,b)\}$.

A closure operation C on $\mathcal{P}(M)$ is called *a nucleus*, if it satisfies: (C4) $C(X) \cdot C(Y) \subseteq C(X \cdot Y)$, for all $X, Y \subseteq M$. Assuming (C1)-(C3), (C4) is equivalent to: for any $C-$closed set X and any $Y \subseteq M$, the sets $Y \backslash X$ and X/Y are $C-$closed [10].

Let $(M, \cdot)$ be a groupoid and C be a closure operation on $\mathcal{P}(M)$. By M_C we denote the family of $C-$closed subsets of M. M_C is closed under infinite meets (intersections), hence it is a complete lattice. For $X, Y \in M_C$ one defines: $X \otimes_C Y = C(X \cdot Y)$ and $X/Y, X \backslash Y$ as above. If C is a nucleus, then M_C with these operations and $\subseteq$ is a residuated groupoid [19].

A phase space $(M, \cdot, R)$ is called *a phase space for* InNL, if $\phi_R = \psi_R$ and the following condition holds:

(Shift) for all $a, b, c \in M$, $R(a \cdot b, c)$ iff $R(a, b \cdot c)$.

It is called a phase space for CNL, if (Shift) holds and R is symmetric.

For any phase space $(M, \cdot, R)$, R is symmetric if and only if $X^{\sim} = X^{-}$ for any $X \subseteq M$. Consequently, if R is symmetric, then $\phi_R = \psi_R$. So every phase space for CNL is a phase space for InNL.

A frame $(M, \cdot, 1, M)$ such that $(M, \cdot, 1)$ is a unital groupoid and $(M, \cdot, R)$ is a phase space is said to be *unital*. If $(M, \cdot, 1, R)$ is a unital phase space, satisfying (Shift), then $R = R_O$ for $O = \{a \in M : R(a, 1)\}$. Also $\{1\}^{\sim} = \{1\}\backslash O = O$, $\{1\}^{-} = O/\{1\} = O$, hence O is ϕ_R-closed and ψ_R-closed. This shows that our notion of a phase space is really more general just for non-unital phase spaces (not considered in the literature on linear logics).

Lemma 1. *Let $(M, \cdot, R)$ be a phase space. (Shift) is equivalent to each of the following conditions:*

(i) $X^{\sim}/Y = X\backslash Y^{-}$ for all $X, Y \subseteq M$,

(ii) $X\backslash Y^{\sim} = (Y \cdot X)^{\sim}$ for all $X, Y \subseteq M$,

(iii) $X^{-}/Y = (Y \cdot X)^{-}$ for all $X, Y \subseteq M$.

Proof. First, (Shift) is equivalent to (i) restricted to one-element sets: for all $a, b \in M$, $\{a\}^{\sim}/\{b\} = \{a\}\backslash\{b\}^{-}$. We have: $x \in \{a\}^{\sim}/b$ iff $x \cdot b \in \{a\}^{\sim}$ iff $R(a, x \cdot b)$. Also: $x \in \{a\}\backslash\{b\}^{-}$ iff $a \cdot x \in \{b\}^{-}$ iff $R(a \cdot x, b)$. So (Shift) is equivalent to: for all $a, b, x \in M$, $x \in \{a\}^{\sim}/\{b\}$ iff $x \in \{a\}\backslash\{b\}^{-}$.

The restricted (i) follows from (i). Also (i) follows from the restricted (i), since for $X = \{a_i\}_{i\in I}$, $Y = \{b_j\}_{j\in J}$, we obtain:

$$X^{\sim}/Y = (\bigcap_{i\in I}\{a_i\}^{\sim})/Y = \bigcap_{i\in I}\bigcap_{j\in J}\{a_i\}^{\sim}/\{b_j\} = \bigcap_{i\in I}\bigcap_{j\in J}\{a_i\}\backslash\{b_j\}^{-} = X\backslash Y^{-}.$$

Here we use the distributive laws for $\cdot, \backslash, /$ and:

$$(\bigcup_{i\in I} Z_i)^{\sim} = \bigcap_{i\in I} Z_i^{\sim},\ (\bigcup_{j\in J} Z_j)^{-} = \bigcap_{j\in J} Z_j^{-}.$$

So (Shift) is equivalent to (i). The remaining equivalences can be proved in a similar way. □

If $\phi_R = \psi_R$, then $M_{\phi_R} = M_{\psi_R}$; we denote this family by M_R. Clearly M_R is closed under $^{\sim}, ^{-}$. We write $\otimes_R$ for $\otimes_{\phi_R}$.

Theorem 1. *Let $(M, \cdot, R)$ be a phase space for InNL. Then, ϕ_R is a nucleus and $(M_R, \otimes_R, \backslash, /, ^{\sim}, ^{-}, \subseteq)$, where the operations are restricted to M_R, is an involutive residuated groupoid.*

Proof. Assume $X \in M_R$, $Y \subseteq M$. There exist $Z, U \subseteq M$ such that $X = Z^{\sim} = U^{-}$. By Lemma 1, $X/Y = U^{-}/Y = (Y \cdot U)^{-}$, hence $X/Y \in M_R$. Similarly, $Y\backslash X = Y\backslash Z^{\sim} = (Z \cdot Y)^{\sim}$, hence $Y\backslash X \in M_R$. Consequently, ϕ_R is a nucleus.

For $X \in M_R$, we have: $X^{-\sim} = \phi_R(X) = X$ and $X^{\sim-} = \psi_R(X) = X$, which yields (DN). (CON) follows from Lemma 1. □

We refer to the algebra M_R, defined above, as *the complex algebra* of the phase space $(M, \cdot, R)$. Let $\mathbf{M} = (M, \cdot, \backslash, /, ^{\sim}, ^{-}, \leq)$ be an involutive residuated groupoid. One defines the (canonical) phase space $(M, \cdot, R)$, by setting: $R(a, b)$ iff $a \leq b^{-}$ (equivalently: $b \leq a^{\sim}$). The mapping $h(a) = \{b \in M : b \leq a\}$ is an embedding of $\mathbf{M}$ into the complex algebra of this canonical phase space. In general, the canonical relation R cannot be represented as R_O for $O \subseteq M$.

An analogous theorem holds for phase spaces for CNL [12]. For a unital phase space $(M, \cdot, 1, R)$, one defines $1_R = \phi_R(\{1\})$. If ϕ_R is a nucleus, then 1_R is a unit for $\otimes_R$. By *a phase space for* InNL1 we mean a unital phase space, satisfying (Shift) and $\phi_R = \psi_R$. It follows from Theorem 1 that for any phase space $(M, \cdot, 1, R)$ for InNL1 the algebra M_R with 1_R and the remaining components as above is a unital involutive residuated groupoid.

Usually, the symmetry of R is easy to verify. It is more difficult to verify $\phi_R = \psi_R$. The following lemma will be used in Section 3; it refines Proposition 1.11 in [1].

Lemma 2. *Let $(M, \cdot, R)$ be a phase space, satisfying (Shift). Let G be a set of generators for M. Then, $\phi_R = \psi_R$ if and only if, for any $x \in G$, the set $\{x\}^{\sim}$ is ψ_R-closed and the set $\{x\}^{-}$ is ϕ_R-closed.*

Proof. The 'only if' part is easy. Assume $\phi_R = \psi_R$. Then, for $X \subseteq M$, $X^{\sim}$ is ϕ_R-closed, hence ψ_R-closed, and X^{-} is ψ_R-closed, hence ϕ_R-closed.

We prove the 'if' part. We show: if X is ϕ_R-closed (resp. ψ_R-closed) and $Y \subseteq M$, then $Y\backslash X$ (resp. X/Y) is ϕ_R-closed (resp. ψ_R-closed). Take $X = Z^{\sim}$. Then $Y\backslash X = (Z \cdot Y)^{\sim}$, by Lemma 1, hence $Y\backslash X$ is ϕ_R =closed. Take $X = Z^{-}$. Then $X/Y = (Y \cdot Z)^{-}$, by Lemma 1, hence X/Y is ψ_R-closed.

Assume the right-hand side of the equivalence. We prove that for any $x \in M$, $\{x\}^{\sim}$ is ψ_R-closed and $\{x\}^{-}$ is ϕ_R-closed. This holds for $x \in G$. Assume that this holds for y, z. We show that this folds for $y \cdot z$. We have: $\{y\}^{\sim} = \psi_R(\{y\}^{\sim})$, hence, using Lemma 1, $\{y \cdot z\}^{\sim} = \{z\}\backslash\{y\}^{\sim} = \{z\}\backslash\{y\}^{\sim\sim-} = \{z\}^{\sim}/\{y\}^{\sim\sim}$. The last set is ψ_R-closed, since $\{z\}^{\sim}$ is so. A similar argument shows that $\{y \cdot z\}^{-}$ is ϕ_R-closed.

We show that every ϕ_R-closed set is ψ_R-closed, and conversely. Let X be ϕ_R-closed. Then $X = Y^{\sim}$ for some Y. We have: $Y^{\sim} = \bigcap_{y \in Y} \{y\}^{\sim}$. Since every set $\{y\}^{\sim}$ is ψ_R-closed, then X is so. The converse can be proved in a similar way.

We have shown $M_{\phi_R} = M_{\psi_R}$. Since $\phi_R(X)$ equals the least ϕ_R-closed set containing X, and similarly for ψ_R, we obtain $\phi_R = \psi_R$. □

This lemma remains true for unital phase spaces. Since the algebra M_R is a complete lattice, we obtain some models for NL, NL1 with additives. If $\cdot$ is associative (commutative), then $\otimes_R$ is so. If $\cdot$ is associative, then (Shift) obviously holds for R_O. If $\cdot$ is commutative and admits 1, then

R_O is symmetrical. Therefore, phase spaces for PLL can be (and have been) defined as quadruples $(M, \cdot, 1, O)$ such that $(M, \cdot, 1)$ is a commutative monoid and $O \subseteq M$.

3 Sequent systems

We present our one-sided sequent system for InNL. Propositional variables are denoted by $p, q, r, \ldots$. Atomic formulas (atoms) are of the form $p^{(n)}$, where p is a variable and $n \in \mathbb{Z}$. One interprets $p^{(n)}$, for $n \geq 0$, as $p^{\sim\cdots\sim}$ with $\sim$ occurring n times, and for $n < 0$, as $p^{-\cdots-}$ with $^-$ occurring $|n|$ times. The connectives are $\otimes, \oplus$.

The elements of the free groupoid generated by all formulas are called *bunches*. The meta-logical notation is like for NL. *Sequents* are the bunches containing at least two formulas. Often we omit the outer parentheses in sequents.

The axioms of our cut-free system are:

$$\text{(id) } p^{(n)}, p^{(n+1)} \text{ for all variables } p \text{ and } n \in \mathbb{Z}.$$

The inference rules are as follows.

$$\text{(r-}\otimes\text{)} \ \frac{\Gamma[(A, B)]}{\Gamma[A \otimes B]}$$

$$\text{(r-}\oplus\text{1)} \ \frac{\Gamma[B] \quad \Delta, A}{\Gamma[(\Delta, A \oplus B)]} \qquad \text{(r-}\oplus\text{2)} \ \frac{\Gamma[A] \quad B, \Delta}{\Gamma[(A \oplus B, \Delta)]}$$

$$\text{(r-shift)} \ \frac{(\Gamma, \Delta), \Theta}{\Gamma, (\Delta, \Theta)}$$

The algebraic models are involutive residuated groupoids. A valuation μ must satisfy: $\mu(p^{(n+1)}) = \mu(p^{(n)})^\sim$, for any atom $p^{(n)}$. μ is extended for sequents by: $\mu((\Gamma, \Delta)) = \mu(\Gamma) \otimes \mu(\Delta)$. We define: $\mathbf{M}, \mu \models \Gamma, \Delta$ iff $\mu(\Gamma) \leq \mu(\Delta)^-$ (equivalently: $\mu(\Delta) \leq \mu(\Gamma)^\sim$).

This system is a dual Schütte style system: the sequent Γ can be written as $\Gamma \Rightarrow$, while in Schütte style systems as $\Rightarrow \Gamma$. We prefer the dual style, since it is more compatible with the syntax of intuitionistic systems (see Section 1). A similar system for Bilinear Logic, i.e. an associative version of InNL1 with additives, was presented by Lambek [26].

We denote this system by S-InNL. Negations $^\sim$,$^-$ are defined in meta-language.

$$(p^{(n)})^\sim = p^{(n+1)} \quad (p^{(n)})^- = p^{(n-1)}$$

$$(A \otimes B)^\sim = B^\sim \oplus A^\sim \quad (A \oplus B)^\sim = B^\sim \otimes A^\sim$$

$$(A \otimes B)^- = B^- \oplus A^- \quad (A \oplus B)^- = B^- \otimes A^-$$

One easily proves $A^{\sim -} = A$, $A^{-\sim} = A$, by induction on A. If μ is a valuation in an involutive residuated groupoid, then:

$$\mu(A^{\sim}) = \mu(A)^{\sim}, \;\; \mu(A^{-}) = \mu(A)^{-}, \text{ for any formula } A. \tag{3}$$

(3) can be proved by induction on A. We write $\vdash \Gamma$, if Γ is provable in S-InNL. By induction on derivations in S-InNL one easily proves that (r-$\otimes$) is reversible: if $\vdash \Gamma[A \otimes B]$ then $\vdash \Gamma[(A, B)]$. Also $\vdash A^{-}, A$ and $\vdash A, A^{\sim}$ for any formula A (use induction on A).

With the cut-rules

$$(\text{cut}^{\sim}) \; \frac{\Gamma[A] \;\; \Delta, A^{\sim}}{\Gamma[\Delta]} \quad (\text{cut}^{-}) \; \frac{\Gamma[A] \;\; A^{-}, \Delta}{\Gamma[\Delta]}$$

S-InNL is strongly complete with respect to involutive residuated groupoids (see Theorem 3).

We want to prove that the cut-rules are admissible in S-InNL by a model-theoretic argument. We need an auxiliary system S_0, which arises from S-InNL, after one has replaced (r-shift) with the following rules:

$$(\text{r-}\oplus 3) \; \frac{A, \Gamma \;\; B, \Delta}{A \oplus B, (\Delta, \Gamma)} \quad (\text{r-}\oplus 4) \; \frac{\Gamma, A \;\; \Delta, B}{(\Delta, \Gamma), A \oplus B}$$

(r-$\oplus$3) is derivable in S-InNL, by (r-$\oplus$2) and (r-shift), and (r-$\oplus$4) is derivable, by (r-$\oplus$1) and (r-shift). By $\vdash_0$ we denote the provability in S_0.

In opposition to an analogous result for CNL in [12], for InNL we need two proof-theoretic lemmas. The complete proofs of these lemmas can be found in [13].

Lemma 3. *The rule (r-shift) is admissible in S_0.*

Proof. We prove: $\vdash_0 \; (\Gamma_1, \Gamma_2), \Gamma_3$ iff $\vdash_0 \; \Gamma_1, (\Gamma_2, \Gamma_3)$. The only-if part is proved by induction on the proof of $(\Gamma_1, \Gamma_2), \Gamma_3$ in S_0, and the converse implication in a similar way. We only consider the case for (r-$\oplus$2).

We consider two subcases. 1°. The active bunch $(A \oplus B, \Delta)$ occurs in Γ_i for some $1 \leq i \leq 3$. We apply the induction hypothesis. 2°. $(A \oplus B, \Delta) = (\Gamma_1, \Gamma_2)$. Then $\Gamma_1 = A \oplus B$, $\Gamma_2 = \Delta$, and the premises are A, Γ_3 and B, Δ. One derives $A \oplus B, (\Delta, \Gamma_3)$, by (r-$\oplus$3). □

Consequently, $\vdash \Gamma$ if and only if $\vdash_0 \Gamma$, for any sequent Γ. We need two new rules.

$$(\text{r-}^{\sim\sim}) \; \frac{A, \Gamma}{\Gamma, A^{\sim\sim}} \quad (\text{r-}^{--}) \; \frac{\Gamma, A}{A^{--}, \Gamma}$$

Lemma 4. *The rules (r-$^{\sim\sim}$) and (r-$^{--}$) are admissible in S-InNL.*

Proof. It suffices to prove that these rules are admissible in S_0. For (r-$^{\sim\sim}$), we prove: if $\vdash_0 D, \Theta$ then $\vdash_0 \Theta, D^{\sim\sim}$, by the outer induction on the number of connectives in D and the inner induction on the proof of D, Θ in S_0. For (r-$^{--}$) the argument is similar.

Assuming our claim for all D' having less connectives than D, we run the inner induction. If D, Θ is an axiom (id), where $D = p^{(n)}$, $\Theta = p^{(n+1)}$, then $\Theta, D^{\sim\sim}$ equals $p^{(n+1)}.p^{(n+2)}$, which is an axiom, too.

Assume that D, Θ is the conclusion of a rule. If D is not the active formula of the rule (this may only happen for (r-$\otimes$), (r-$\oplus$1), (r-$\oplus$2)), then one applies the (inner) induction hypothesis to the premise which contains D (this must be $\Gamma[B]$ in (r-$\oplus$1) and $\Gamma[A]$ in (r-$\oplus$2)), next applies the same rule.

Assume that D is the active formula of the rule. This may happen for (r-$\otimes$) and (r-$\oplus$3). We only consider (r-$\otimes$). So $D = A \otimes B$ and the premise is $(A, B), \Theta$. By Lemma 3 and the outer induction hypothesis, we obtain $\vdash_0 A, (B, \Theta)$, $\vdash_0 (B, \Theta), A^{\sim\sim}$, $\vdash_0 B, (\Theta, A^{\sim\sim})$, $\vdash_0 (\Theta, A^{\sim\sim}), B^{\sim\sim}$, $\vdash_0 \Theta, (A^{\sim\sim}, B^{\sim\sim})$, hence $\vdash_0 \Theta, D^{\sim\sim}$, by (r-$\otimes$). □

As a consequence, $\vdash A^-, \Gamma$ if and only if $\vdash \Gamma, A^\sim$; we write $\Gamma \Rightarrow A$ for A^-, Γ. We define:

$$[A] = \{\Gamma : \vdash \Gamma \Rightarrow A\}.$$

A sequent Γ is said to be *valid* in InNL, if $\mathbf{M}, \mu \models \Gamma$ for all involutive residuated groupoids $\mathbf{M}$ and all valuations μ in $\mathbf{M}$. We prove that S-InNL is weakly complete.

Theorem 2. *For any sequent* Γ, $\vdash \Gamma$ *if and only if* Γ *is valid in InNL.*

Proof. It is easy to verify that the axioms (id) are valid and all rules preserve validity (in fact, they preserve the truth for μ in $\mathbf{M}$). We only consider (r-shift) in the top-down direction. Assume $\mathbf{M}, \mu \models (\Gamma, \Delta), \Theta$. Then $\mu(\Gamma) \otimes \mu(\Delta) \leq \mu(\Theta)^-$. So $\mu(\Delta) \leq \mu(\Gamma) \backslash \mu(\Theta)^- = \mu(\Gamma)^\sim / \mu(\Theta)$. We obtain $\mu(\Delta) \otimes \mu(\Theta) \leq \mu(\Gamma)^\sim$, which yields $\mathbf{M}, \mu \models \Gamma, (\Delta, \Theta)$. Hence the only-if part holds.

For the 'if' part, we construct a counter-model for any unprovable sequent. We consider a phase space $(M, \cdot, R)$ such that $(M, \cdot)$ is the free groupoid generated by the set of formulas (so $\Gamma \cdot \Delta = (\Gamma, \Delta)$) and: $R(\Gamma, \Delta)$ iff $\vdash \Gamma, \Delta$. Due to (r-shift), this phase space satisfies (Shift).

For any formula A, we have: $[A] = \{\Gamma : R(A^-, \Gamma)\} = \{\Gamma : R(\Gamma, A^\sim)\}$, which yields:

$$[A] = \{A^-\}^\sim = \{A^\sim\}^-, \ [A^\sim] = \{A\}^\sim, \ [A^-] = \{A\}^-. \tag{4}$$

Consequently, $[A]$ is ϕ_R-closed and ψ_R-closed, for any formula A. We obtain $\phi_R = \psi_R$, by Lemma 2. So $(M, \cdot, R)$ is a phase space for InNL.

By Theorem 1, the algebra $(M_R, \otimes_R, \backslash, /, {}^{\sim}, {}^{-}, \subseteq)$ is an involutive residuated groupoid. We define a valuation μ.

$$\mu(p) = [p] \text{ where } p = p^{(0)}$$

$$\mu(p^{(n+1)}) = \mu(p^{(n)})^{\sim} \text{ for } n \geq 0, \;\; \mu(p^{(n-1)}) = \mu(p^{(n)})^{-} \text{ for } n \leq 0$$

By induction on the number of connectives in A, we prove:

$$A \in \mu(A) \subseteq [A] \text{ for any formula } A\,. \tag{5}$$

$A = p^{(n)}$. We proceed by induction on $|n|$. For $n = 0$, we have $\mu(p) = [p]$ and $p \in [p]$, since $p, p^{\sim}$ is an axiom. Assume (5) for $n \geq 0$. From $\mu(p^{(n)}) \subseteq [p^{(n)}] = \{p^{(n+1)}\}^{-}$ (use (4)) we obtain $\{p^{(n+1)}\}^{-\sim} \subseteq \mu(p^{(n)})^{\sim} = \mu(p^{(n+1)})$ (use (3)), and $p^{(n+1)} \in \phi_R(\{p^{(n+1)}\}) = \{p^{(n+1)}\}^{-\sim}$, hence $p^{(n+1)} \in \mu(p^{(n+1)})$. From $\{p^{(n)}\} \subseteq \mu(p^{(n)})$ we obtain $\mu(p^{(n+1)}) \subseteq \{p^{(n)}\}^{\sim} = [p^{(n+1)}]$ (use (3), (4)). In a similar way, assuming (5) for $n \leq 0$, we obtain (5) for $n - 1$.

$A = B \otimes C$. We need the following property.

(P1) If $X \in M_R$ and $(B, C) \in X$, then $B \otimes C \in X$.

Assume that $X \in M_R$ and $(B, C) \in X$. Fix Y such that $X = Y^{\sim}$. For all $\Gamma \in Y$, $\vdash \Gamma, (B, C)$, hence $\vdash \Gamma, B \otimes C$, by (r-$\otimes$). So $B \otimes C \in X$.

By the induction hypothesis, (5) holds for B and C. From $B \in \mu(B)$ and $C \in \mu(C)$ we obtain $(B, C) \in \mu(B) \cdot \mu(C) \subseteq \mu(B) \otimes \mu(C) = \mu(B \otimes C)$, hence $B \otimes C \in \mu(B \otimes C)$, by (P1). Also $\mu(B) \subseteq [B] = \{B^{-}\}^{\sim}$ and $\mu(C) \subseteq [C] = \{C^{-}\}^{\sim}$. Hence, for all $\Gamma \in \mu(B)$, $\vdash B^{-}, \Gamma$, and for all $\Delta \in \mu(C)$, $\vdash C^{-}, \Delta$. By (r-$\oplus$3), $\vdash C^{-} \oplus B^{-}, (\Gamma, \Delta)$, for all $\Gamma \in \mu(B)$, $\Delta \in \mu(C)$. Consequently, $\mu(B) \cdot \mu(C) \subseteq \{(B \otimes C)^{-}\}^{\sim} = [B \otimes C]$, which yields $\mu(B \otimes C) = \phi_R(\mu(B) \cdot \mu(C)) \subseteq [B \otimes C]$, since $[B \otimes C]$ is ϕ_R−closed.

$A = B \oplus C$. Since B and B^{-} have the same number of connectives, and similarly for C and C^{-}, then (5) holds for B^{-} and C^{-} by the induction hypothesis. So $C^{-} \otimes B^{-} \in \mu(C^{-} \otimes B^{-}) \subseteq [C^{-} \otimes B^{-}]$. We obtain $[C^{-} \otimes B^{-}]^{\sim} \subseteq \mu(C^{-} \otimes B^{-})^{\sim} \subseteq \{C^{-} \otimes B^{-}\}^{\sim}$, which yields (5) for $B \oplus C$, since $B \oplus C \in \{B \oplus C\}^{-\sim} = [C^{-} \otimes B^{-}]^{\sim}$, $\mu(C^{-} \otimes B^{-})^{\sim} = \mu(B \oplus C)$ and $\{C^{-} \otimes B^{-}\}^{\sim} = [B \oplus C]$. This finishes the proof of (5).

Assume $\nvdash \Gamma, \Delta$. From $A \in \mu(A)$, for any formula A, one easily proves $\Theta \in \mu(\Theta)$ for any bunch Θ, by induction on the number of commas in Θ. So $\Gamma \in \mu(\Gamma)$ and $\Delta \in \mu(\Delta)$. By the assumption, $\Gamma \notin \mu(\Delta)^{-}$, hence $\mu(\Gamma) \not\subseteq \mu(\Delta)^{-}$. Consequently, Γ, Δ is not valid. □

Corollary 1. *The rules (cut$^{\sim}$), (cut^{-}) are admissible in S-InNL.*

Proof. Both rules preserve the truth for μ in $\mathbf{M}$, hence the validity. □

This model-theoretic proof of the cut-elimination theorem, similar to the proofs for associative linear logics in [23, 32], is not constructive. A constructive, syntactic proof is given in [13].

Theorem 3. *S-InNL with (cut$^\sim$), (cut$^-$) is strongly complete with respect to involutive residuated groupoids.*

Proof. Let Φ be a set of sequents. With the cut-rules, every sequent Γ, Δ is deductively equivalent to a sequent A, B; A arises from Γ by replacing each comma by $\otimes$, and so for B and Δ. We assume that all sequents in Φ are of the latter form.

Now, $\vdash$ denotes the provability in S-InNL with the cut-rules. Notice that (r-$^{\sim\sim}$), (r-$^{--}$) are derivable. For (r-$^{\sim\sim}$), we derive $\Gamma, A^{\sim\sim}$ from $A^\sim, A^{\sim\sim}$ and A, Γ, by (cut$^-$).

We consider a phase space $(M, \cdot, R)$, defined as in the proof of Theorem 2 except that: $R(\Gamma, \Delta)$ iff $\Phi \vdash \Gamma, \Delta$. We define $[A] = \{\Gamma : \Phi \vdash \Gamma \Rightarrow A\}$, where $\Gamma \Rightarrow A$ is as above. The equations (4) hold. Accordingly, $(M, \cdot, R)$ is a phase space for InNL. We consider the involutive residuated groupoid M_R and the valuation μ, defined there. One proves (5) and a stronger claim:

$$\mu(A) = [A] \text{ for any formula } A. \tag{6}$$

We need the following property.

(P2) If $X \in M_R$ and $A \in X$, then $[A] \subseteq X$.

Assume that $X \in M_R$ and $A \in X$. Fix Y such that $X = Y^\sim$. Then, $\Phi \vdash \Gamma, A$ for all $\Gamma \in Y$, and consequently, $\Phi \vdash \Gamma, \Delta$ for all $\Gamma \in Y$, $\Delta \in [A]$ (use (cut$^-$)). So $[A] \subseteq X$.

Accordingly, $A \in \mu(A)$ yields $[A] \subseteq \mu(A)$. Hence (5) entails (6).

If $(A, B) \in \Phi$, then $[A] \subseteq [B^-]$ (use (cut$^-$), (r-$^{--}$)), hence $\mu(A) \subseteq \mu(B)^-$, by (6), (3). If $\Phi \nvdash \Gamma, \Delta$, then $\mu(\Gamma) \not\subseteq \mu(\Delta)^-$, as in the proof of Theorem 2. □

The sequent system S-InNL1 is an extension of S-InNL. We add the empty bunch ϵ and constants 1 and 0. Sequents are all nonempty bunches. We add one new axiom and two new rules.

$$\text{(a.0) } 0 \quad \text{(r-1) } \frac{\Gamma[\Delta]}{\Gamma[(1, \Delta)]} \quad \frac{\Gamma[\Delta]}{\Gamma[(\Delta, 1)]}$$

We assume $(\epsilon, \Gamma) = \Gamma = (\Gamma, \epsilon)$. In rules (r-$\oplus$1), (r-$\oplus$2) we admit $\Delta = \epsilon$, and similarly for (cut$^\sim$), (cut$^-$). In the derivable rules (r-$\oplus$3), (r-$\oplus$4) both Γ and Δ may be empty; in (r-$^{\sim\sim}$), (r-$^{--}$) Γ may be empty.

The algebraic models are unital involutive residuated monoids. We define: $\mathbf{M}, \mu \models \Gamma$ iff $\mu(\Gamma) \leq 0$. In metalanguage $1^\sim = 1^- = 0$, $0^\sim = 0^- = 1$.

All results, proved above and appropriately modified, are true for S-InNL1. The proofs are quite similar, but they need certain modifications (more cases).

These systems can be augmented with additive connectives $\wedge, \vee$, interpreted as meet and join in lattice-ordered (l.o.) involutive residuated groupoids. All results of Sections 2 and 3 remain true for these extensions. In a similar way, one can prove these results for associative (commutative) logics of this kind.

4 InNL versus InNL(k)

Let a be an element of an involutive residuated groupoid. We use the notation $a^{(n)}$ in the same meaning as in Section 3, e.g. $a^{(2)} = a^{\sim\sim}$, $a^{(-2)} = a^{--}$. Let k be a positive integer. An involutive residuated groupoid is said to be $k-$*cyclic*, if $a^{(k)} = a$ for any $a \in M$. If k is odd, then $a \leq b$ entails $b^{(k)} \leq a^{(k)}$. So for an odd k, $\leq$ is the identity relation and $a \otimes b = b \oplus a$ in any $k-$cyclic involutive residuated groupoid. In what follows we assume that k is even. Observe that $a^{-} = a^{(k-1)}$ in $k-$cyclic algebras of this kind.

InNL(k) is InNL from Section 1 with the new axiom $A^{(k)} \Leftrightarrow A$. This logic is strongly complete with respect to $k-$cyclic involutive residuated groupoids. Clearly InNL(2) amounts to CNL.

We introduce a sequent system S-InNL(k). The atoms are of the form $p^{(n)}$ for $0 \leq n < k$. All axioms and rules are as for S-InNL, but n, $n+1$ are computed modulo k. So the axioms are: $p^{(n)}, p^{(n+1)}$, for $0 \leq n < k-1$ and $p^{(k-1)}, p^{(0)}$. We assume $p^{(0)} = p$.

The metalanguage negations $^{\sim}, ^{-}$ are defined as in Section 3 (we compute $n+1$ and $n-1$ modulo k). All syntactic equations from Section 3 remain true. Also $A^{(k)} = A$, since k is even. We also obtain (3) for any valuation in a $k-$cyclic involutive residuated groupoid. Lemmas 3 and 4 remain true; now S_0 arises from S-InNL(k) like the former S_0 from S-InNL.

Theorem 4. *S-InNL(k) is weakly complete with respect to $k-$cyclic involutive residuated groupoids.*

Proof. The whole proof of Theorem 2 can be repeated, except that we cannot prove (at this moment) that the algebra $(M_R, \otimes_R, \backslash, /, ^{\sim}, ^{-}, \subseteq)$ is $k-$cyclic. Let $\mathbf{M}_R^{\mu}$ be the subalgebra whose universe consists of $\mu(A)$ for all formulas A. Clearly $\mathbf{M}_R^{\mu}$ is an involutive residuated groupoid and μ is a valuation in $\mathbf{M}_R^{\mu}$. By (3), $\mu(A)^{(k)} = \mu(A^{(k)}) = \mu(A)$, hence $\mathbf{M}_R^{\mu}$ is $k-$cyclic. If $\not\vdash \Gamma, \Delta$, then $\mathbf{M}_R^{\mu}, \mu \not\models \Gamma, \Delta$, and consequently, Γ, Δ is not valid. □

Accordingly, (cut$^{\sim}$), (cut^{-}) are admissible in S-InNL(k). With these rules S-InNL(k) is strongly complete with respect to $k-$cyclic involutive residuated groupoids. Again the proof of Theorem 3 can be repeated, and

we consider $\mathbf{M}_R^\mu$ as above. Consequently, S-InNL(2) is a cut-free system for CNL, different from those in [22, 12]. It can be shown that M_R in the proof of Theorem 4 is $k-$cyclic (use the infinite De Morgan laws for $^\sim$ in M_R).

We return to S-InNL. For any bunch Γ and $m \in \mathbb{Z}$, by $I(\Gamma, m)$ we denote the bunch arising from Γ, after one has replaced each atom $p^{(n)}$ by $p^{(n+m)}$.

Lemma 5. *For any sequent Γ and $m \in \mathbb{Z}$, Γ is provable in S-InNL if and only if $I(\Gamma, m)$ is provable in S-InNL.*

Proof. Assume that Γ is provable in S-InNL. In a proof of Γ substitute $p^{(n+m)}$ for $p^{(n)}$, for any atom $p^{(n)}$. We obtain a proof of $I(\Gamma, m)$. This yields the only-if part and entails the 'if' part, since $\Gamma = I(I(\Gamma, m), -m)$. □

Let Γ be a sequent, and let m be the greatest positive integer n such that some atom $p^{(-n)}$ occurs in Γ; if there is none, we set $m = 0$. The provability of Γ is equivalent to the provability of $I(\Gamma, m)$, which contains no $p^{(n)}$ with $n < 0$; such a sequent is said to be $^--$*free*.

Lemma 6. *Let Γ be a $^--$free sequent. Let m be the greatest integer n such that some atom $p^{(n)}$ occurs in Γ. Let $k > m + 1$. Then, Γ is provable in S-InNL if and only if Γ is provable in S-InNL(k).*

Proof. Assume that Γ is provable in InNL. Then Γ is valid in InNL, and consequently, in $k-$cyclic involutive residuated groupoids. By Theorem 4, Γ is provable in S-InNL(k).

Assume that Γ is provable in S-InNL(k). There exists a (cut-free) proof of Γ such that every formula in this proof is a subformula of a formula in Γ. Consequently, no atom $p^{(k-1)}$ appears in this proof, hence no axiom $p^{(k-1)}, p$ is used. This proof is a proof in S-InNL. □

Therefore, if a sequent Γ is not valid in InNL, then it is not valid in some $k-$cyclic involutive residuated groupoid. This result for InGL was obtained in [18] with the aid of a two-sided sequent system for InGL. Our proof seems simpler and works for this richer logic as well.

InNL(k) has similar properties as CNL (which amounts to InNL(2)):

(T1) an interpolation property: if $\Phi \vdash \Gamma[\Delta]$, where $\Gamma[\Delta] \neq \Delta$, then there exists $D \in T$ (an interpolant of Δ) such that $\Phi \vdash \Gamma[D]$ and $Phi \vdash \Delta \Rightarrow D$, where T is the closure of the set of formulas appearing in $\Phi \cup \{\Gamma[\Delta]\}$ under subformulas and $^\sim$ (T is finite, if one uses S-InNL(k)),

(T2) the strong finite model property: the sequents provable in InNL(k) from a finite set Φ coincide with those which follow from Φ in finite models,

(T3) the languages generated by the type grammars based on InNL(k) are precisely the ($\epsilon-$free) context-free languages,

(T4) the (finitary) consequence relation for InNL(k) is P-TIME.

(T1)-(T4) can be proved quite similarly as their particular cases for CNL in [12]. We skip the details. Let us only note some essential points.

Let k be fixed. Given a finite set of formulas T, its closure under subformulas and $^\sim$ can be computed in polynomial time (in the size of T and k), since one applies $^\sim$ $k-1$ times to each formula; we denote this closure by $\bar{T}$.

(T1) can be proved by induction on proofs in S-InNL(k). If Δ is a formula, then $D = \Delta$. Let us only consider the rule (r-shift): top-down, with premise $(\Gamma, \Delta'), \Theta$ and conclusion $\Gamma, (\Delta', \Theta)$, where $\Delta = (\Delta', \Theta)$. Let D' be an interpolant of Γ in the premise. Then $D^\sim$ is an interpolant of (Δ', Θ) in the conclusion.

(T2) can be proved like Theorem 3 for S-InNL(k) except that M is the free groupoid generated by $\bar{T}$, where T is the set of formulas appearing in $\Phi \cup \{\Gamma\}$. We only consider $\bar{T}-$proofs, i.e. the proofs in S-InNL(k) which employ formulas from $\bar{T}$ only. We write $\Phi \vdash^{\bar{T}} \Gamma$, if there exists a $\bar{T}-$proof of Γ from Φ. We define $[A]_{\bar{T}} = \{\Gamma \in M : \Phi \vdash^{\bar{T}} \Gamma\}$, and replace $[A]$ by $[A]_{\bar{T}}$ in the whole reasoning. We obtain $\mu(A) = [A]_{\bar{T}}$ for any $A \in \bar{T}$. Thus, if $\Phi \vdash^{\bar{T}} \Gamma$ does not hold, then Γ is not true for μ. Although the resulting frame is infinite, its complex algebra is finite. By (T1), there exist only finitely many sets of the form $\{\Gamma\}^\sim$, since $\{\Gamma\}^\sim$ equals the union of sets $[D]_{\bar{T}}$ for some formulas $D \in \bar{T}$. Consequently, there are only finitely many closed sets, hence M_R is finite.

(T3) follows from (T1). Let $G = (\Sigma, I, A_0)$ be a type grammar based on InNL(k). Let T denote the set of all formulas involved in I plus A_0. By (T1), every $\bar{T}-$sequent $\Gamma \Rightarrow A$, provable in InNL(k) can be proved on the basis of provable $\bar{T}-$sequents of the form $(A_1, A_2) \Rightarrow B$ and $B \Rightarrow C$ (restricted sequents) with the aid of (NL-cut). This yields a context-free grammar equivalent to G. Its terminal symbols are those of G, the nonterminal symbols are the formulas from $\bar{T}$, the production rules are the reversed, restricted $\bar{T}-$sequents provable in InNL(k), and A_0 is the start symbol. It is known that every $\epsilon-$free context-free language is generated by a type grammar based on NL, and NL can be replaced by InNL(k), since InNL(k) is a strongly conservative extension of NL.

Finally, (T4) holds, since all restricted $\bar{T}-$sequents, provable in InNL(k), can be proved in S-InNL(k), limited to restricted $\bar{T}-$sequents; see the proof for CNL in [12].

From (T2)-(T4) we infer the main results of this section. Analogous results can be obtained for InNL1.

(T5) InNL possesses the finite model property.

(T6) The type grammars based on InNL generate the $\epsilon-$free context-free languages.

(T7) InNL is P-TIME.

REMARK. This proof of P-TIME complexity assumes that the size of $p^{(n)}$ is defined as $|n| + 1$, where n denotes the absolute value of n. This is reasonable, if $|n|$ is small (which is the case for types appearing in linguistic applications) or $p^{(n)}$ is understood as a metalanguage abbreviation of p with $|n|$ negations. Another proof, given in [13], yields a polynomial algorithm with $|n|$ defined as the length of the binary (or decimal) representation of n.

Accordingly, a context-free grammar equivalent to the given type grammar G based on InNL can be constructed in polynomial time in the size of G. On the contrary, an analogous construction for the type grammars based on L (due to Pentus [33]) is exponential; no polynomial construction exists, if the hypothesis P=NP is false. This follows from the fact that L is NP-complete [34].

5 Final comments

We briefly discuss the significance of our results for type grammars.

Since a leitmotive of type grammars is a reduction of parsing to formal proofs in appropriate logical calculi, independent of the particular language, this theory develops and studies different logics suitable for this purpose. This research is located on the borderline of logic and linguistics: logical ideas are applied in grammar and, conversely, linguistics stimulates new logical formalisms. Linear logics belong to this area. This paper contributes to nonassociative linear logics, which have not been much worked out earlier. Our interest in these logics is motivated by their close relations to Nonassociative Lambek Calculus and other systems of this kind.

InNL and CNL are strongly conservative extensions of NL. In Section 1, we have already explained that this allows us to use the former instead of the latter in type grammars. If one translates NL-types into InNL-types in the formalism of S-InNL, then formulas with negations appear. For example, $pn\backslash s$ is translated as $pn^{\sim} \oplus s$ and $(pn\backslash s)/pn$ as $(pn^{\sim} \oplus s) \oplus pn^{-}$. If $(n\backslash n)/(s/pn)$ is assigned to 'whom' in 'girl whom John admires', this type is translated as $(n^{\sim} \oplus n) \oplus (pn^{--} \otimes s^{-})$. Types of the latter form were used in e.g. [14, 15]. In pregroup grammars $\otimes$ and $\oplus$ collapse; one writes AB for both $A \otimes B$ and $A \oplus B$, A^r for $A^{\sim}$ and A^l for A^{-}. So the type of 'whom' is written as $n^r n[pn]^{ll} s^l$ (product is associative).

Due to conservativity, our results for InNL, CNL are stronger than their analogues for NL: the former entail the latter, but not conversely. For example, from the fact that InNL-grammars are equivalent to context-free grammars it follows that NL-grammars are so (for NL-grammars, this was known earlier). Interestingly, these conservativity results also hold for associative (commutative) linear logics versus their intuitionistic fragments, but it was not known until [2] has appeared. ([12] shows a simpler, general

method for such conservativity proofs with the aid of phase spaces.) Certainly, the authors of [32] did not know it. They developed intuitionistic phase spaces in order to prove the finite model property for intuitionistic linear logics, which was earlier proved by Lafont [23] for MALL (with a claim that this proof also works for noncommutative logics). Intuitionistic phase spaces are interesting for themselves, but the main results of [32] (the finite model property) simply follow from Lafont's results, by conservativity.

Logics with negation(s) exhibit interesting symmetries (dualities), absent in their intuitionistic fragments. For instance, the application law $A \otimes (A\backslash B) \Rightarrow B$ and the co-application law $B \Rightarrow (B \otimes A)/A$ are related by the duality: $a \leq b$ iff $b^- \leq a^-$.

Why the consequence relation is relevant in type grammars? At this point, the present author differs in opinions from most linguists working in type grammar. They, as a rule, adhere to *radical lexicalism*: the whole language-dependent grammatical information has to be contained in the type lexicon (i.e. the map I), and parsing is to be done within a pure logic. In some situations, it seems reasonable to admit parsing within *a theory*, i.e. a logic augmented with nonlogical assumptions (axioms). Lambek [27] used such assumptions in pregroup grammars, e.g. $s_1 \Rightarrow s$, $s_2 \Rightarrow s$, where s is the type of statement, s_1 of statement in present tense, s_2 of statement in past tense, and many other arrows of this kind (poset arrows). One can consider L, NL etc. augmented with the assumptions corresponding to the production rules of a fixed context-free grammar. Then, the type logic can infer interesting consequences. For instance, L transforms the given context-free grammar into an equivalent basic categorial grammar [8]. One can approximate a stronger logic by a weaker, but more efficient, logic. In particular, L (which is NP-complete) can be approximated by NL augmented with finitely many sequents provable in L (but not closed under substitution). For instance, $(pn\backslash s)/pn \Leftrightarrow pn\backslash(s/pn)$ is provable in L, but not in NL. One can add this particular law to NL, without assuming the general pattern $(A\backslash B)/C \Leftrightarrow A\backslash(B/C)$ (which yields the associative law for $\otimes$), just for a more flexible treatment of transitive verb phrases in the nonassociative framework. Since the finitary consequence relation for NL is P-TIME [9], this approximation leads to an efficient parsing procedure. The same holds for Cyclic MLL versus CNL. For InNL, the complexity of the consequence relation is not known.

The operations $X^\sim$ and X^-, defined in Section 2, are a special case of polarities $X^\triangleright$ and $Y^\triangleleft$, considered in lattice theory. For $R \subseteq U \times V$, $X \subseteq U$, $Y \subseteq V$, one defines:

$$X^\triangleright = \{v \in V : \forall_{u \in X}\, R(u,v)\}, \quad Y^\triangleleft = \{u \in U : \forall_{v \in Y}\, R(u,v)\}.$$

This yields a Galois connection and two closure operations ${}^{\triangleright\triangleleft}$ and ${}^{\triangleleft\triangleright}$; let us denote them by ψ and ϕ. One obtains two dual (complete) lattices: the

first consists of $\psi-$closed subsets of U and the second of $\phi-$closed subsets of V (so-called *concept lattices*).

Clark [17] considered *syntactic concept lattices*, where $U = \Sigma^*$, $V = \Sigma^* \times \Sigma^*$ (the set of contexts) and, for a fixed language $L \subseteq \Sigma^*$, one defines:

$$R_L(u, (v_1, v_2)) \text{ iff } v_1 u v_2 \in L\,.$$

The $\psi-$closed subsets of Σ^* are interpreted as the syntactic categories determined by the language L (the main category). In fact, the lattice of syntactic categories is a residuated lattice with $\backslash, /$ defined as in the algebra of languages. Syntactic concept lattices are models of Full Lambek Calculus. It is easy to adapt this construction for languages of phrase structures, as processed by nonassociative type grammars.

Accordingly, algebras of closed sets, defined similarly as in Section 2, possess sound linguistic interpretations. In our phase spaces $U = V$, and we impose additional constraints upon R: (Shift), $\phi = \psi$. The relation R_L, defined as above (for nonassociative contexts), need not satisfy them, if L is, say, the set of declarative sentences of a natural language, represented as phrase structures. Nonetheless, every nonassociative syntactic concept lattice can be isomorphically embedded into a lattice-ordered involutive residuated groupoid (this follows from the strong conservativity). A linguist, working with InNL instead of NL, may interpret the formulas of InNL in the extended model. This resembles the mathematical construction of complex numbers, extending real numbers. Although complex numbers do not possess such intuitive interpretations as reals (e.g. as lengths, weights etc.), they give rise to a more regular and deeper mathematical theory.

At the end, let us point out evident connections with classical modal logics. It is well-known that $\otimes, \backslash, /$ can be treated as binary modal operators. We focus on linear negations. Given a modal frame (M, R), where $R \subseteq M^2$, and $X \subseteq M$one defines $\Diamond(X) = \{u \in M : \exists_{v \in X} R(u, v)\}$, $\Box(X) = (\Diamond(X^c))^c$ (here X^c denotes the complement of X), and similarly $\Diamond^{\downarrow}, \Box^{\downarrow}$ with respect to the converse relation $R^{\cup}$. These operations satisfy the unary residuation laws: $\Diamond(X) \subseteq Y$ iff $X \subseteq \Box^{\downarrow}(Y)$, and $\Diamond^{\downarrow}(X) \subseteq Y$ iff $X \subseteq \Box(Y)$ (so $\Diamond, \Box^{\downarrow}$ and $\Diamond^{\downarrow}, \Box$ form residuation pairs).

For a phase space, $(M, \cdot, R)$, we have $X^{\sim} = \Box^{\downarrow}(X^c)$, $X^{-} = \Box(X^c)$, where the modalities are defined for (M, R^c). Consequently $\phi_R(X) = \Box^{\downarrow}\Diamond(X)$, $\psi_R(X) = \Box\Diamond^{\downarrow}(X)$. The condition $\phi_R = \psi_R$ can be expressed by the modal axiom $\Box^{\downarrow}\Diamond p \Leftrightarrow \Box\Diamond^{\downarrow} p$, where $\Leftrightarrow$ stands for biconditional of classical logic. Also $p^{\sim} \Leftrightarrow \Box^{\downarrow}\neg p$, $p^{-} \Leftrightarrow \Box\neg p$, where $\neg$ stands for classical negation. (Shift) can be expressed by $p^{\sim}/q \Leftrightarrow p\backslash q^{-}$.

References

[1] Abrusci, V.M.: Phase semantics and sequent calculus for pure noncommutative classical linear propositional logic. Journal of Symbolic Logic 56: 1403–1451 (1991).

[2] Abrusci, V.M.: Classical Conservative Extensions of Lambek Calculus. Studia Logica 71.3: 277–314 (2002).

[3] Ajdukiewicz, K.: Die syntaktische Konnexität. Studia Philosophica 1: 1–27 (1935).

[4] Bar-Hillel, Y.: A quasi-arithmetical notation for syntactic description. Language 29: 47–58 (1953).

[5] Bar-Hillel, Y., Gaifman, C., and Shamir, E.: On categorial and phrase structure grammars. Bull. Res. Council Israel F 9: 155–166 (1960).

[6] Bastenhof, A.: Categorial Symmetry. Ph.D. Thesis, University of Utrecht (2013).

[7] van Benthem, J.: Language in Action. Categories, Lambdas and Dynamic Logic. North-Holland (1991).

[8] Buszkowski, W.: Extending Lambek Grammars to Basic Categorial Grammars. Journal of Logic, Language, and Information 5: 279–295 (1996).

[9] Buszkowski, W.: Lambek Calculus with Nonlogical Axioms. In: Casadio, C., Scott, P.J., and Seely, R.A.G. (eds.), Language and Grammar. CSLI Lecture Notes 168, pp. 77–93, CSLI Publications (2005).

[10] Buszkowski, W.: Interpolation and FEP for logics of residuated algebras. Logic Journal of IGPL 19: 437–454 (2011).

[11] Buszkowski, W.: Some Syntactic Interpretations in Different Systems of Full Lambek Calculus. In: Ju, S., Liu, H., and Ono, H. (eds.), Modality, Semantics and Intepretations, Logic in Asia: Studia Logica Library, pp. 23–48, Springer (2015).

[12] Buszkowski, W.: On Classical Nonassociative Lambek Calculus. In: Amblard, M,, de Groote, Ph., Pogodalla, S., and Retoré, C. (eds.), Logical Aspects of Computational Linguistics, Lecture Notes in Computer Science, vol. 10054, pp. 68–84, Springer (2016).

[13] Buszkowski, W.: Involutive Nonassociative Lambek Calculus: Sequent Systems and Complexity. Bulletin of The Section of Logic. To appear.

[14] Casadio, C.: Non-Commutative Linear Logic in Linguistics. Grammars 4.3: 167–185 (2001).

[15] Casadio, C. and Lambek J.: A Tale of Four Grammars. Studia Logica 71.3: 315–329 (2002).

[16] Chvalovsky, K.: Undecidability of consequence relation in full nonassociative Lambek calculus. Journal of Symbolic Logic 80: 567–576 (2015).

[17] Clark, A.: A learnable representation for syntax using residuated lattices. In: de Groote, Ph., Egg, M., and Kallmeyer, L. (eds.), Formal Grammar, LNCS 5591, pp. 183–198, Springer (2011).

[18] Galatos, N. and Jipsen, P.: Residuated frames with applications to decidability. Transactions of American Mathematical Society 365: 1219–1249 (2013).

[19] Galatos, N., Jipsen, P., Kowalski, T., and Ono, H.: Residuated Lattices. An Algebraic Glimpse at Substructural Logics. Elsevier (2007).

[20] Girard, J.-Y.: Linear logic. Theoretical Computer Science 50: 1–102 (1987).

[21] Grishin, V.N.: On a generalization of the Ajdukiewicz-Lambek system. In: Studies in Non-classical Logics and Formal Systems, pp. 315–343, Nauka, Moscow (1983).

[22] de Groote, Ph. and Lamarche, F.: Classical Non-Associative Lambek Calculus. Studia Logica 71.3: 355–388 (2002).

[23] Lafont, Y.: The finite model property of various fragments of linear logic. Journal of Symbolic Logic 62: 1202–1208 (1997).

[24] Lambek, J.: The mathematics of sentence structure. The American Mathematical Monthly 65: 154–170 (1958).

[25] Lambek, J.: On the calculus of syntactic types. In: Jakobson, R. (ed.) Structure of Language and Its Mathematical Aspects, pp. 166–178. AMS, Providence (1961).

[26] Lambek, J.: Cut elimination for classical bilinear logic. Fundamenta Informaticae 22: 53–67 (1995).

[27] Lambek, J.: Type Grammars Revisited. In: Lecomte, A., Lamarche, F., and Perrier, G. (eds.), Logical Aspects of Computational Linguistics, LNAI 1582, pp. 1–27, Springer (1999).

[28] Moortgat, M.: Categorial Type Logic. In: van Benthem, J. and ter Meulen, A. (eds.), Handbook of Logic and Language, pp. 93–177, Elsevier, MIT Press, Amsterdam (1997).

[29] Moortgat, M.: Symmetric Categorial Grammar. Journal of Philosophical Logic 38(6): 681–710 (2009).

[30] Morrill, G.: Type Logical Grammar. Categorial Logic of Signs. Kluwer (1994).

[31] Moot, R, and Retoré, C.: The Logic of Categorial Grammars. LNCS 6850, Springer (2012).

[32] Okada, M. and Terui, K.: The finite model property for various fragments of intuitionistic linear logic. Journal of Symbolic Logic 64: 790–802 (1999).

[33] Pentus, M., Lambek grammars are context-free. Proc. 8 IEEE Symposium on Logic in Computer Science, pp. 429–433 (1993).

[34] Pentus, M.: Lambek calculuc is NP-complete. Theoretical Computer Science 357: 186–201 (2006).

[35] Yetter, D.N.: Quantales and (non-commutative) linear logic. Journal of Symbolic Logic 55: 41–64 (1990).

Combining Machine Learning and Semantic Features in the Classification of Corporate Disclosures

Stefan Evert[1], Philipp Heinrich[1], Klaus Henselmann[2], Ulrich Rabenstein[3], Elisabeth Scherr[2], and Lutz Schröder[3]

[1]Dept. Germanistik und Komparatistik, FAU Erlangen-Nürnberg
[2]School of Business and Economics, FAU Erlangen-Nürnberg
[3]Dept. of Computer Science, FAU Erlangen-Nürnberg

Abstract

We investigate an approach of improving statistical text classification by combining machine learners with an ontology-based identification of domain-specific topic categories. We apply this approach to ad hoc disclosures by public companies. This form of obligatory publicity concerns all information that might affect the stock price; relevant topic categories are governed by stringent regulations. Our goal is to classify disclosures according to their effect on stock prices (negative, neutral, positive). In the feasibility study reported here, we combine natural language parsing with a formal background ontology to recognize disclosures concerning a particular topic, viz. retirement of key personnel. The semantic analysis identifies such disclosures with high precision and recall. We then demonstrate that machine learners benefit from the additional ontology-based information in different prediction tasks.

1 Introduction

With the amount of electronically available information rising, there is increasing interest in developing new means of assessing the semantics of corporate disclosures in order to better handle the high information load. Prior research successfully explores the use of textual analysis to predict stock performance (Bollen et al. 2010; Verchow 2011; Jegadeesh and Wu 2013; Ding et al. 2015), often based on "big data" sources such as Twitter trends. Although our use case also involves the prediction of an econometric variable, accurate prediction of stock prices is *not* our main goal. The broader aim of our work is to extract hidden information from financial texts; we therefore do not make use of additional data sets such as social media to enhance the performance of our task solvers.

In the feasibility study reported here, we aim to improve the performance of a statistical text classifier by integrating knowledge retrieved from the text by an ontology-based reasoner. Our use case is the prediction of stock market reactions after the publication of corporate events according to German law. In so-called *ad hoc disclosures*, companies

have to report any important event that might affect the stock price. Although the relevant types of events are essentially predefined by law,[1] this information is not indicated explicitly in the disclosures and has to be derived from the textual content.

Ad hoc disclosures are a suitable object of the investigation of combining machine learning with ontological reasoning for two reasons: Firstly, these disclosures are supposed to provide information relevant to the stock price and thus offer a straightforward task for machine learning and evaluation (the prediction of stock prices from text). Secondly, companies have an incentive to downplay negative events and hide them "between the lines". Aiming to reveal hidden indicators in the ad hoc disclosures, we focus entirely on their textual content and do not make use of external information from social media or other sources. This makes the prediction task fairly hard and it is thus astonishing that our trained classifiers provide an effective trading strategy. Except for Verchow (2011), who aims at analyzing capital market efficiency and whose unsophisticated computational linguistic methodology leads to rather poor results, we are not aware of any prior work that attempts stock market prediction from ad hoc disclosures.

2 Methodology

The present methodological section is structured as follows: Section 2.1 gives an overview of our corpus, the target variable and the associated prediction task. Section 2.2 briefly introduces the machine learning techniques and their evaluation; section 2.3 motivates the necessity of creating an ontology and outlines its design. Section 2.4 concludes the methodological part by explaining the different ways of integrating machine learners with ontological features[2]; this section also presents further evaluation techniques for the combination of machine learning (ML) with ontological information.

2.1 Data basis and prediction tasks

Corpus We use a sample of announcements of corporate events provided by the DGAP service of the Equity Story AG. Our sample selection starts with over 80,000 mandatory announcements of material events that have been disseminated via the DGAP between mid-1996 and mid-2012. We restrict our analyses to those disclosures that are written in English[3] and that are machine-readable. Due to these constraints and further restrictions on available metadata (see the following paragraph), we obtain a final corpus of 28,287 documents ("textual units") such as the following example:

> Montabaur, December 31, 2001. Michael Scheeren, CFO of United Internet AG and with the company for 11 years, will retire from his position on the Executive Board as of December 31, 2001. It is planned that he will replace Mr. Hans-Peter

[1] See the guideline issued by the Federal Financial Supervisory Authority (BaFin) (2009) for a list of possible price-sensitive events.

[2] We call statistical classifiers and their ontologically enhanced versions more generally "task solvers".

[3] German law requires the material event disclosures to be in German, in another accepted language or in English depending on specific criteria.

> Bachmann on the Supervisory Board from January 1, 2002. Scheeren will retain his close ties to the Group as he remains Chairman of the Supervisory Boards of AdLINK AG, 1&1 Internet AG and twenty4help AG. He will also represent United Internet AG on the Supervisory Boards of GMX AG, jobpilot AG and NTplus AG. Mr. Norbert Lang has been named as successor for Michael Scheeren. Lang has been with United Internet since 1994. After first heading the financial department, he joined the United Internet Executive Board one year ago.

Target variable For each ad hoc disclosure i, we measure its effect on the stock market using an event study following prior literature (Strong 1992; McWilliams and Siegel 1997; Corrado 2011). In particular, the market model is used to calculate the market-adjusted stock return surrounding the disclosure date t of the material event:

$$AR_{it} = R_{it} - E(R_{it}) = R_{it} - (\hat{\alpha}_i + \hat{\beta}_i \cdot R_{Mt}) \quad (1)$$

Daily market-adjusted returns or abnormal returns (AR_{it}) are calculated as the deviation between the observed stock return of each individual company (R_{it}) and the expected stock return ($E(R_{it})$). We use the return of the CDAX index as a proxy for the market return and estimate $E(R_{it})$ by regressing a historic series of observed daily stock returns (R_{it}) on the corresponding daily market returns (R_{Mt}) using ordinary least squares (OLS) estimation. The estimation period starts 6 days ($t-6$) and spans up to 155 days ($t-155$) prior to the event date. The estimated intercept ($\hat{\alpha}_i$) and slope ($\hat{\beta}_i$) of the OLS model are then inserted into equation (1) to calculate the abnormal return (AR_{it}).

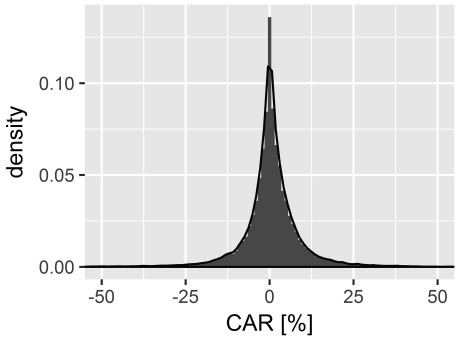


Figure 1: Distribution of the target variable CAR in the corpus (excluding outliers with $|CAR_{it}| > 50$).

We use daily return index data from Thomson Reuters Datastream that is adjusted for capital events (e.g., dividends, stock splits); daily returns are calculated as logarithmic returns (i.e. as $\log(V_f/V_i)$ where V_i is the initial and V_f the final value; use of this quantity is standard to ensure symmetry).[4] In order to account for the fact that part of the information relating to the event is priced early or late, we use an event window of three trading days. Hence, the cumulative abnormal return (CAR_{it}) surrounding each event announcement date (t) is calculated as the sum of the abnormal returns between one day prior ($t-1$) and one day after ($t+1$) the disclosure of the event. The distribution of the target variable is heavy-tailed, slightly skewed, and concentrates around 0 (cf. Figure 1).

Prediction tasks Although the target variable is metric, we abstain from regression analysis for two reasons: Firstly, the data shows heavy tails, which makes it difficult

[4] Non-trading days are excluded.

for regressors to find suitable weights. Secondly, it is far more important in practice to distinguish between positive and negative reactions than to predict the exact degree of the reaction. We hence set ourselves the prediction task of recognizing negative, neutral and positive responses based on a ternary categorization (1/3 of disclosures each); the categories are constructed by means of the respective quantiles of the empirical distribution of CAR.

Since these artificially created categories are hard to distinguish for machine learners – especially if the true CAR value is close to a category boundary – we also analyze the performance of the task solvers in a slightly modified prediction task with more clear-cut categories, i.e. ternary categorization into well-separated categories (20% of most negative and most positive reactions and the 20% closest to the median). We thus refer to the first one of these tasks as the *difficult* prediction task and to its modified version as the *easy* one (see Table 1 for an overview).

	negative	neutral	positive	corpus size
difficult	9,433	9,436	9,418	28,287 (100%)
retirements	*413*	*341*	*292*	*1046 (3.7%)*
easy	5,661	5,645	5,648	16,954 (60%)
retirements	*267*	*205*	*167*	*639 (3.8%)*

Table 1: The two prediction tasks to be solved by the machine learning classifiers and their combinations with ontological features. The **easy** prediction task is a slight modification of the **difficult** one, involving more sharply separated categories. The rows labeled *retirements* show the number of disclosures concerning key personnel turnover in each category (cf. section 2.3).

2.2 ML classification

Our ML classifiers for solving the prediction tasks in table 1 are based on a simple bag-of-words feature set (FM1, cf. section 2.4) with tf.idf weighting.[5] After heuristic deletion of boilerplate footers and headers, removal of stop words, e-mail addresses, URLs, punctuation and numbers, as well as lower-casing and lemmatization, the resulting feature vocabulary contains $n_{voc} = 32{,}401$ lemmas.

We present results for Multinomial Naïve Bayes (MNB) and Logistic Regression (MaxEnt) with $\ell 1$-penalty tuned by 10-fold cross-validation on the training set (for implementation details see Pedregosa et al. 2011). Other machine learning algorithms such as Support Vector Machines and a modified MNB used by Verchow (2011) yielded similar results.

[5]Preliminary experiments including longer n-grams and part of speech tags in the feature matrices did not lead to consistently higher performance.

Evaluation of the ML approach We use accuracy in 10-fold stratified cross-validation (90% training, 10% test data) as a performance measure and compute 95% confidence intervals for the mean accuracy across all 10 folds (based on a normal approximation). Since we have equally-sized categories and stratify the class distribution in the cross-validation,[6] a random baseline classifier achieves an accuracy of $1/3 = 33.3\%$ in our ternary classification tasks.

In order to demonstrate the practical usefulness of our ML approach, we also evaluate the machine learning classifier by means of a simple trading strategy: (1) *buying* if the ML approach predicts category *positive*, (2) *short selling* if it yields *negative*, and (3) *holding* if the result is *neutral*. A scalar performance measure is given by the sum of all individual net gains of CAR values.[7] In this setting, we use constant classifiers that always make the same decision as baselines.

2.3 Ontological feature extraction

Our idea is to use the semantic event categories that regulate the emission of disclosures in the first place in order to improve the ML classifiers. Recall that the disclosures are sent out for very specific reasons, but these are not explicitly mentioned in the text of a disclosure or in the associated metadata. Although the boundaries between different event categories are somewhat fuzzy, most of the disclosures are sent out for one particular reason: manual analysis of a sample of 1,000 disclosures showed that only about 15% fall into more than one topic category.

Motivation for ontological feature extraction The background information about the initial reason to send out the disclosures is valuable and provides a different sort of knowledge than the sort of "semantic information" that can be retrieved from the text itself by unsupervised learning (e.g. automatic clustering of the disclosures). Techniques such as Latent Dirichlet Allocation (LDA) or Latent Semantic Indexing are often found to be helpful in text classification because they reduce the high-dimensional bag-of-words feature space (with $n_{voc} = 32{,}401$ dimensions in our case) to a comparatively small number of *latent semantic* dimensions. Machine learners are expected to perform better because information is packaged more densely into the latent features and a smaller number of parameters needs to be trained. Exploratory tests showed that our ML classifier does not benefit from such dimensionality reduction techniques, though.

We might also use the latent semantic information to pre-classify the disclosures into meaningful categories, viz. the pre-defined set of approximately 40 topics regulating their emission. As a matter of fact, the most prominent topic in our corpus (according to an LDA model) with a mean latency of almost 25%[8], is made up of rather generic lemmas

[6] That is to say: all categories contain equal numbers of disclosures in each fold of the cross-validation.

[7] The trading strategy rests on the assumption that we can buy or sell the shares after the material event and thus indeed collect the net gain of CAR values.

[8] The result of an LDA is a probability vector for each document comprising the probabilities with which each of the topics has contributed to the creation of the document. The "mean latency" of a topic is thus the average probability of that topic across all documents.

such as

> *product, service, technology, lead, position, agreement, new, future, work, solution, system, customer, provide, production, subsidiary, focus, ceo, industry, develop,* and *management.*

The topic above could e.g. be interpreted as *future products and contracts* or alike, yet there are clearly ambiguous and noisy terms such as *ceo*, *industry*, etc., which make the interpretation very speculative. The second most prominent topic with more than 19% mean latency contains the following lemmas:

> *earnings, previous, tax, ebit, figure, quarter, compare, profit, revenue, net, positive, income, first, month, rise, increase, operating, ebitda, result, fiscal.*

This topic points towards quarterly reports. However, neither of the topics point towards a clearly recognizable reason for the emission of a disclosure.[9] Furthermore, the distribution of LDA topics on the particulary interesting subset of disclosures that inform about retirements (see the following section 2.3) is almost identical to their distribution on the full corpus. The topics that can be retrieved from an LDA analysis thus do not help in recognizing particular topics that can be identified manually.

We thus develop a formal ontology to retrieve meaningful semantic features. Since this is expensive with regards to implemenation effort, we concentrate on one frequent and particularly interesting category, namely disclosures concerned with the retirement of key personnel.

NLP pre-processing The first step of operationalizing the corporate texts is a preprocessing stage in which disclosures are analyzed using various off-the-shelf natural language processing techniques, including part-of-speech tagging, morphological analysis, named entity recognition, syntactic parsing, coreference resolution and word sense disambiguation.

The Stanford CoreNLP suite (Manning et al. 2014) offers publicly available tools for the first five tasks. They are part of a pipeline architecture, i.e. every component can access the results of the previous components. For word sense disambiguation, we use the algorithm described in Banerjee and Pedersen (2002) and the sense inventory of the lexical semantic database WordNet (Miller 1995). In the ontological representation, the disambiguated words are mapped to WordNet concepts (*synsets*[10]). The ontology consists of three components:[11]

- A TBox capturing relations among concepts, essentially obtained by extracting relevant information from WordNet for the terms encountered in the text.
- A manually maintained TBox capturing domain-specific background knowledge.

[9] See also Feuerriegel et al. (2015) for a more thorough analysis of the semantic space of ad hoc disclosures.

[10] *Synsets* are sets of synonyms representing a lexical semantic concept or word sense.

[11] *ABox* and *TBox* are the assertion and terminological components of the ontology, respectively.

- An ABox recording the content of the parsed disclosures, generated from the NLP results.

We discuss these parts in more detail below.

Ontology creation from NLP results We first describe the automatically generated parts of the ontology. It has to be emphasized that this ontology is not learned in any sense; rather, the procedure is essentially aimed at transforming linguistically analyzed texts into the Web Ontology Language (OWL), additionally taking into account lexical semantic information from WordNet. WordNet information is provided in terms of a chain of subclass or subproperty inclusions connecting the word form actually appearing in a text to its synset identified by the word sense disambiguation module. E.g. for the word form *leaves* (possibly indicating a retirement event) this takes the following shape (Listing 1):

ObjectProperty: leave

ObjectProperty: leaves
  SubPropertyOf: leave

ObjectProperty: 2383440_Leave_depart_pull_up_stakes
  SubPropertyOf: leave

Listing 1: OWL representation of WordNet information for word form *leaves*.

The last, most specific object property relates to the synset corresponding to the relevant sense of *leaves*. It is composed of the synset's unique WordNet identifier (2383440), followed by the list of all synonyms in the set (to ensure human readability).

As indicated above, the NLP results are transformed into an ABox. The default procedure is to map subjects and objects of sentences, identified by the dependency analysis, to individuals in the ABox, whereas the verbs connecting subjects and objects become object properties. For prepositional objects, the preposition is made part of the object property, which is then named in the form ⟨verb⟩_⟨preposition⟩. If the auxiliary verb *will* is detected in connection with the verb (e.g. if the disclosure states that the CEO *will* resign rather than that he has already resigned), the object property is named announced_⟨X⟩, where X is the original name of the object property, and marked as being a subproperty of both announced and X. Subjects and objects receive as a type the concept generated from their synset according to the word sense disambiguation, and receive as facts their mutual relationship as specified by the synset of the verb. For example, the sentence *John Doe leaves the company* with the syntactic dependency analysis in Figure 2 is translated into the ABox depicted in Listing 2.

Note that each syntactic dependency connects only two words. For compound nouns, the rightmost noun is regarded as the head noun, and the other component nouns are linked to the head noun via a *compound* relation. Compound nouns have to be recomposed from the syntactic dependencies, which results in the individual *John Doe* rather than just *Doe*. Coreferences are resolved while creating the ontology, so facts referring to a pronoun are attached to the corresponding individual.

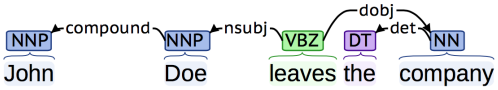


Figure 2: Dependency parse of *John Doe leaves the company.*

Individual: John_Doe
Types: Person
Facts: 2383440_leave_depart_pull_up_stakes company

Individual: company
Types: 8058098_Company

Listing 2: ABox representation of sentence *John Doe leaves the company.*

In this case, the types of the individuals are inferred from named entity recognition (Person) and morphological analysis (Company). Appositions are also used to infer types: the dependent of an apposition determines an additional type for its governor, and triples describing the dependent are assigned to the governor. Prepositional triples are prefixed by the dependent of the apposition. For instance, from the phrase *John Doe, CFO of the company*, one obtains the dependency relations

$$\text{appos}(\textit{Doe}, \textit{CFO}) \quad \text{and} \quad \text{of}(\textit{CFO}, \textit{company}),$$

which extend the knowledge about John Doe in the way depicted in Listing 3.

The previous examples always contained the main piece of information, e.g. on someone leaving a company, or doing something in general, in a subject-predicate-object-like structure. A sentence like "He announced the retirement of John Doe" does not fit into this pattern. Therefore our system uses derivational relations from WordNet to transform triples like *of(retirement, John Doe)* into a subject-predicate-object structure, *retire(John Doe, dummy)*. The dummy individual is needed because the intransitive verb *retire* (from which *retirement* is derived) does not take an object. This type of normalization simplifies querying the assertional knowledge parsed from the text in subsequent steps.

Background knowledge The system is supported by a static, manually maintained background ontology capturing general and domain knowledge that is not explicit in the

Individual: John_Doe
Types: Person, CFO
Facts: 2383440_leave_depart_pull_up_stakes company.
CFO_of company

Listing 3: Extended ABox for sentence *John Doe leaves the company.*

Class: 9916601_chief_financial_officer_cfo
EquivalentTo: works_on **some** Cfo_position
SubClassOf: works_on **exactly** 1 Executive_board_position

Class: Cfo_leave1
EquivalentTo: leave **some** Cfo_position,
Cfo **and** leave **some** Executive_board_position

Class: Cfo_leave2
EquivalentTo: Cfo **and** (leave **some** Executive_board),
leave **some** Cfo_position

Class: leave3
EquivalentTo: (have **some** (Contract **and** expire **some** owl:Thing)),
SubClassOf: leave **some** Position

Class: leave4
EquivalentTo: agree **some** (Termination **and** (of **some** Mandate)),
SubClassOf: leave **some** Position

Class: leave5
EquivalentTo: submit **some** Resignation,
SubClassOf: leave **some** Position

Listing 4: Excerpt from the background ontology.

text of the disclosures. Some of the relevant facts are quite simple, e.g. that stepping down is a form of leaving and that "Executive Board" and "Management Board" are synonyms. Other axioms are more interesting and capture combinations of standard jargon with basic knowledge of the domain. E.g. at the domain-specific level we include axioms saying that CFOs work on exactly one executive board position, and that they retire from their CFO position iff they retire from their executive board position.[12] At a less specific level there are axioms saying that, e.g., letting your contract expire, agreeing to the termination of your mandate, and submitting your resignation all amount to leaving your current position. The formulation of statements such as these is illustrated in Listing 4.

Querying With the ontology in place, we can now detect disclosures concerning retirement of key personnel by querying the ABox generated from the disclosure for persons leaving from something. The formulation of the corresponding query is shown in Listing 5. The filter statements serve to eliminate multiple results that differ only in the value of the *?leave* variable, i.e. use different subproperties of *leave* but refer to the same person and position. That is, the query is set up in such a way as to return only the triples with the most specific object property as the instantiation for *?leave*.

In case the query returns any result, the ad hoc disclosure is marked as containing

[12]The universal validity of these axioms may be debatable but since OWL does not incorporate default reasoning, there appears to be no realistic way to ensure stricter accuracy.

```
SELECT DISTINCT ?person ?leave ?object WHERE{
    ?person ?leave ?object.
    ?person a :Person.
    ?leave rdfs:subPropertyOf :leave.
    FILTER NOT EXISTS{ ?person ?leave2 ?object.
        ?leave2 rdfs:subPropertyOf ?leave.
        FILTER NOT EXISTS{?leave2 owl:equivalentProperty ?leave. }}}
```

Listing 5: The main query used to detect leaving persons.

a message about a retirement. In case the instantiation of *?leave* is a subproperty of *announced*, the disclosure is additionally annotated as being (only) an announcement.

Evaluation of the ontological approach The ontological detection of retirement events and announcements among all ad hoc disclosures was tested on a set of 300 messages containing any inflected form of the words *leave* or *retire*. The disclosures were categorized manually as retirement (178 messages) or non-retirement (122 messages). The low baseline accuracy of 59.3% shows that the mere occurrence of the keywords *leave* and *retire* is not a reliable predictor. Our algorithm obtained recall and precision values of 90.4% and 97% for retirement events, respectively.[13] Regarding the additional property of retirements being only announced rather than already realized, 75 of 139 messages were successfully identified as (only) announcing at least one retirement, and 6 were falsely classified as (mere) announcements (recall 54%, precision 92.5%). It is, of course, not entirely surprising that automated detection of the event ("retirement") as such works better than automated detection of the much more abstract question of its factuality. Subsequent to these tests, we ran the ontological classifier on the whole dataset of 28,287 disclosures, obtaining a set of 1,046 disclosures (ca. 3.7%) classified as retirement events.

2.4 Integration methods

We now turn to equipping the ML classifiers with the ontologically extracted retirement feature in order to improve their performance. Our idea is that the ontological information about the types of material events that regulate the dissemination of the disclosures in the first place can be used for splitting the overall problem into smaller sub-problems: A machine learner trained solely on retirement disclosures is confronted with an easier task than a system that does not have any information about the reasons for the dissemination of the disclosures at hand; just as a human expert confronted with very specific disclosures has an easier task than someone who is confronted with an unstructured bulk of disclosures.

Our first combination of ML and ontology is by means of adding a single "retirement" feature to the document-lemma feature matrix (**FM2**). However, since a single retirement

[13] 161 of the 178 true retirement messages were detected correctly by the algorithm (true positives) while 5 disclosures were incorrectly marked as retirements (false positives).

feature can easily be overseen amongst other features, we experiment with a separation of the vocabulary of the retirement disclosures from the vocabulary of non-retirements: If a disclosure is recognized as a retirement by the ontological model, the string `retirement` is appended to each lemma in the text (**FM3**).

This method has the disadvantage that the ML classifier cannot generalize information about the general meaning of lemmas (e.g. *risk* or *losses*) gathered from the much larger remainder of the corpus to the retirement disclosures. It is likely to underperform in this setting because it is effectively restricted to a small training corpus. We thus consider a third combination method that mirrors the retirement vocabulary (**FM4**): All retirement disclosures now retain the original lemmas, but are complemented with an *additional* retirement vocabulary. In the example disclosure above, lemmas such as $Montabaur_{\text{retirement}}$, $CFO_{\text{retirement}}$, and $company_{\text{retirement}}$ are added without deleting the original lemmas.

To put it in other words, the reasoning behind separate and mirrored vocabularies is as follows: A *single* retirement feature might not be recognized efficiently by a machine learner. A *separate* vocabulary, moreover, discriminates against retirement disclosures, since the machine learner cannot exploit features from the much larger non-retirement part of the training corpus for the retirements; as a result, the amount of training data for lemmas in the retirement vocabulary is drastically reduced. A separate vocabulary is equivalent to two separate machine learners being trained for the retirement disclosures and the non-retirements, respectively. Last but not least, the retirement features are weight *adjustments* in the case of the mirrored vocabulary: Here the machine learner can learn features both on retirements and non-retirements and can then exploit this knowledge for all disclosures. Including the basic feature matrix (**FM1**), there are thus four feature matrices that can be used for prediction (see Table 2 for an overview).

features	*description*	n_{voc}
FM1	vanilla feature matrix without retirement feature	32,401
FM2	FM1 with a single *additional retirement feature*	32,402
FM3	FM1 with a *separate vocabulary* for the retirement disclosures	37,652
FM4	FM1 with a *mirrored vocabulary* for the retirement disclosures	38,762

Table 2: The four different feature matrices used for prediction.

Evaluation of integration methods Since the integration methods essentially differ in feature matrices, they can be compared to the original classifiers (and their respective baselines) in a straightforward manner by means of accuracy in 10-fold cross-validation. Moreover, for retirement disclosures, another baseline is readily at hand: Since the retirement feature is weakly yet significantly associated with the target variable, retirement disclosures can be classified *ontologically*: The greater part of retirement disclosures lead to a negative stock market reaction (cf. Table 1), so that the ontology already outperforms the baseline by assigning the category *negative* to all retirement disclosures.

3 Results and Discussion

There are two kinds of effects to be analyzed: Firstly, the effect of ontological information on the prediction quality of task solvers can be quantified. Secondly, one can observe how the feature weights are affected by the additional information, which gives interesting insights into the different usage of language in particular discourse topic domains.

3.1 Prediction results

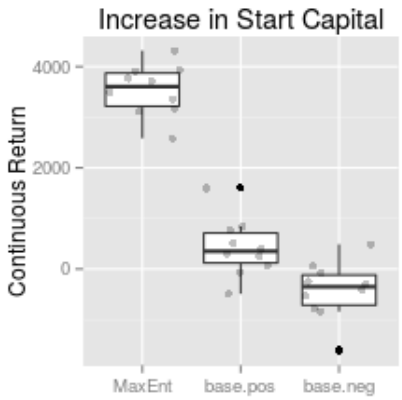


Figure 3: Profit made by simple trading strategy based on classification of disclosures in the easy prediction task (task solver: MaxEnt using FM1).

The complete results for the four prediction tasks can be found in Tables 3 and 4 for the hard and the easy prediction task, respectively. The left hand tables show performance on the whole corpus, the right hand table considers solely retirement disclosures.[14] One row of a table shows the performance of the different classifiers for given feature matrices. In most cases, accuracy values are higher for MaxEnt than for MNB. Comparing the different prediction tasks with one another, one can unsurprisingly see that performance is higher in the case of clear-cut categories. Generally, the machine learners consistently outperform the random baseline (33.3%) and the ontological baseline (which varies for each split into training and test corpus since we do not stratify the number of retirement disclosures in the cross-validation).

For instance, MaxEnt with feature set FM1 significantly outperforms the 33.3% baseline in the easy prediction task with an accuracy of **51.9%** (± **1.9%**). Moreover, Figure 3 shows substantial net gain when using MaxEnt with FM1 and applying the trading strategy outlined in section 2.2: Compared to the baseline traders which always opt for *sell* (base.neg) and *buy* (base.pos), respectively, the accumulated continuous returns of the task solver result in a mean profit of more than 0.2% per disclosure and a considerable increase of our start capital despite the simple approach.

More interestingly, one column of a table allows for comparison between the different feature matrices. FM4 (mirrored vocabulary) consistently outperforms all the other feature matrices. The effect is higher on the sub-corpus of retirements for the reasons elaborated above. In the case of the retirement disclosures, it is worth mentioning that the ontological baseline (predicting class *negative* for all retirement disclosures, given at the bottom lines of the right hand panels of the performance tables 3 and 4) presents

[14] Since less than 4% of the disclosures deal with key personnel turnover in all prediction tasks, "zooming" onto the subcorpus of retirement disclosures is more likely to reveal the effect of including ontological features.

difficult	*full*		*retirements*	
	MNB	MaxEnt	MNB	MaxEnt
FM1	.437 (±.016)	.456 (±.018)	.395 (±.101)	.426 (±.092)
FM2	.437 (±.016)	.456 (±.016)	.395 (±.101)	.426 (±.092)
FM3	.437 (±.015)	.456 (±.019)	.429 (±.124)	.414 (±.091)
FM4	.439 (±.014)	**.459** (±.025)	.445 (±.106)	**.450** (±.128)
baseline	1/3 = .333		.396 (±.086)	

Table 3: Performance (mean accuracy and 95% confidence interval) of the task solvers in the difficult prediction task on the whole corpus (left panel) and on the subcorpus of retirement disclosures (right panel), using different feature matrices (FM1–4). The naïve baseline (majority classifier) is given by 1/3 = 33.3%, an improved (ontological) baseline is given for the retirement corpus.

easy	*full*		*retirements*	
	MNB	MaxEnt	MNB	MaxEnt
FM1	.485 (±.024)	**.519** (±.019)	.467 (±.126)	.479 (±.115)
FM2	.485 (±.024)	.518 (±.014)	.467 (±.123)	.481 (±.117)
FM3	.482 (±.022)	**.519** (±.021)	.431 (±.090)	.470 (±.098)
FM4	.486 (±.022)	**.519** (±.018)	.477 (±.111)	**.500** (±.092)
baseline	1/3 = .333		.419 (±.087)	

Table 4: Performance of the task solvers in the easier prediction task (with well-defined categories). Performance increases consistently as in the difficult prediction task when mirroring the vocabulary (FM4). A separate vocabulary (FM3) decreases performance, a single retirment feature (FM2) does not change performance at all.

itself as a strong competitor for the machine learners. Statistical learning is, however, in almost all cases better than this ontological baseline.

3.2 Feature weight analysis

We identified lemmas whose feature weights are substantially different in retirement disclosures than in non-retirements (which can be seen from FM3), or which obtain a relatively high "adjustment" weight in the mirrored retirement vocabulary (in the best-performing feature set FM4). Table 5 shows such lemmas based on feature weights for the category *positive*. Results for category *negative* are omitted since they show similar patterns.

For example, the lemmas *exceed* (1.293 for category *positive* in FM1) and *improvement* (0.708 in FM1) are generally associated with a positive CAR response. However, FM4 reduces these weights in retirement disclosures by −0.019 for *exceed* and −0.014 for *improvement*, showing that they imply a different outcome for this event type (see

section 2.4 for an explanation of weight adjustments).[15] The relatively small adjustement is probably due to the low overall proportion of retirements, and the effect becomes much clearer with a separate vocabulary in FM3: the feature weights on retirement disclosures are −0.021 (*exceed*) and −0.018 (*improvement*), respectively, showing that these lemmas no longer indicate a positive stock market reaction. Similarly, the lemma *insolvency* is generally associated with a negative reaction, but indicates a positive response when used in retirement disclosures.

	FM1	**FM3**		**FM4**	
lemma		non-ret.	ret.	non-ret.	ret.
exceed	1.293	1.293	-0.021	1.293	-0.019
fall	-0.864	-0.842	-0.034	-0.855	-0.027
career	0.090	-0.033	0.115	0.044	0.089
improvement	0.708	0.696	-0.018	0.700	-0.014
rise	0.612	0.616	-0.024	0.614	-0.023
weak	-0.769	-0.766	-0.012	-0.769	-0.009
lower	-1.022	-1.012	-0.041	-1.018	-0.028
positive	1.149	1.130	-0.007	1.137	-0.015
insolvency	-0.386	-0.447	0.081	-0.417	0.059

Table 5: Lemmas whose feature weights for category *positive* are substantially different in retirement disclosures (additional feature weights in FM3 and FM4) from their overall feature weights (FM1).

4 Conclusion

Machine learners are used in many prediction tasks of computational linguistics. We have combined a semantics-based approach to recognition of message content with a machine-learning classification of documents, specifically of corporate disclosures according to their effect on the stock price.

Machine learners benefit from ontological information since they can thus deal with a more specific realm of language use. The core idea tested in our feasibility study is that words are used more consistently within the specific domain of retirement disclosures. The effect on prediction accuracy is small, yet consistent. Testing for statistical significance by use of a McNemar test shows that some of the improvements are indeed significant.

Future work will be aimed partly at refining the ontological approach to improve its precision and recall (both already above 90% on the main target feature, retirements, but

[15] FM2 is omitted here because its lemma feature weights are almost identical to FM1. Recall that FM2 just adds a single feature indicating retirement disclosures (with a correct indication of category *negative*), so it cannot account for the differences in language use between retirements and other messages that we are interested in here.

not achieving comparable performance for more fine-grained information such as detecting the position that the retiree steps down from). On the other hand, the results of our study make it seem likely that broadening the ontological model to recognize additional features (e.g. patents granted, loans taken up) will further improve the prediction accuracy for the eventual target, the effect of a disclosure on the stock price. Last but not least, we will strive to develop new methods for exploiting the subjective use of language in different domains in order to improve prediction accuracy.

References

Satanjeev Banerjee and Ted Pedersen. An adapted Lesk algorithm for word sense disambiguation using WordNet. In *Proceedings of the Third International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing '02, pages 136–145, London, UK, 2002. Springer-Verlag.

Johann Bollen, Huina Mao, and Xiao-Jun Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8, October 2010.

C. Corrado. Event studies: A methodology review. *Accounting and Finance*, 51:207–234, 2011.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Deep learning for event-driven stock prediction. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (ICJAI)*, pages 2327–2333, 2015.

Federal Financial Supervisory Authority (BaFin). Issuer guidelines, 2009.

Stefan Feuerriegel, Antal Ratku, and Dirk Neumann. Which News Disclosures Matter? News Reception Compared Across Topics Extracted from the Latent Dirichlet Allocation. *News Reception Compared Across Topics Extracted from the Latent Dirichlet Allocation (February 13, 2015)*, 2015.

N. Jegadeesh and D. Wu. Word power: A new approach for content analysis. *J. Financial Economics*, 110:712–729, 2013.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

A. McWilliams and D Siegel. Event studies in management research: Theoretical and empirical issues. *Academy of Management J.*, 40:626–657, 1997.

George A Miller. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39, 1995.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, October 2011.

N. Strong. Modelling abnormal returns: A review article. *J. Business Finance & Accounting*, 19:533–553, 1992.

Thomas Verchow. *Ad-hoc-Publizität und Kapitalmarkteffizienz: Eine Untersuchung basierend auf der Textanalyse von Ad-hoc-Mitteilungen.* PhD thesis, Ulm University, Faculty of Mathematics and Economics, 2011.

Relational Nouns, Inverse Linking and Haddock Descriptions: a unified dependent type account

Justyna Grudzińska and Marek Zawadowski

Overview In this paper [1], we argue for a unified dependent type analysis of two puzzling phenomena: inverse linking ([10], [7]) and Haddock descriptions ([6], [3], [2]). Inverse linking constructions (ILCs) refer to the syntactic environments in which the embedded quantifier phrase (QP) takes scope over the embedding one. For example, sentence (1) *A representative of every country missed a meeting* can be understood to mean that a different representative of each country missed a potentially different meeting in each case, i.e., it allows a reading in which *every country* outscopes *a representative*. This poses a puzzle for standard scope-assignment strategies, for there is independent evidence that scoping out of DP islands should be disallowed. For example, sentence (2) *Two politicians spy on someone from every city* cannot be understood to mean that for every city c, there are two politicians who spy on someone from c, i.e., it does not have a reading in which *two politicians* takes scope in between the two nested QPs. The second phenomenon to be discussed in our talk relates to the so-called Haddock descriptions. Haddock (relative) descriptions, e.g. *the rabbit in the hat*, pose a difficulty for standard presuppositional accounts of definite descriptions, for they carry a kind of 'polyadic presupposition' that there is a single pair $\langle x, y\rangle$ such that x is a hat, y is a rabbit, and y is in x, rather than the standard presuppositions that there be a unique rabbit and a unique hat. Thus *the rabbit in the hat* can be used felicitously in the context with multiple salient hats and multiple salient rabbits, as long as there is exactly one pair of a rabbit and a hat such that the rabbit is in the hat. To tackle the two puzzles, we shall develop a new dependent type account from the perspective of a semantic system combining generalized quantifiers ([1], [11]) with dependent types ([9], [8]).

Semantics with dependent types Dependencies are ubiquitously used and interpreted by natural language speakers. Our semantics with dependent types has proved successful in accounting for a number of natural language phenomena (studied by us so far) in which dependencies play a crucial role (see [4], [5]). The two key features of our approach are:

[1] This work builds on the paper accepted to the TbiLLC 2017 conference.

many-sorted (many-typed) analysis and type dependency. Our analysis is many-sorted in the sense that it includes many basic types (e.g. type *M(onth)*, type *W(oman)*, ...). The variables of our semantic system are always typed. Types are interpreted as sets. In a system with many types, types can depend on the variables of other types. One example of such a dependence of types is that if m is a variable of the type of months M, there is a type $D(m)$ of the days in that month: $m : M, d : D(m)$. If we interpret type M as a set $\|M\|$ of months, then we can interpret type D as a set of the days of the months in $\|M\|$, i.e. as a set of pairs $\|D\| = \{\langle a, k\rangle : k \text{ is (the number of) a day in month } a\}$ equipped with the projection $\pi_{D,m} : \|D\| \to \|M\|$. The particular sets $\|D\|(a)$ of the days of the month a can be recovered as the *fibers* of this projection (the preimages of $\{a\}$ under $\pi_{D,m}$) $\|D\|(a) = \{d \in \|D\| : \pi_{D,m}(d) = a\}$. **Relational nouns** Two kinds of common nouns can be distinguished: sortal (e.g. *woman*) and relational (e.g. *representative*). Whereas in the Montagovian setting sortal CNs are interpreted as one-place relations (expressions of type $\langle e, t\rangle$), in our dependent type theoretical framework they are treated as types. For example, *woman* is interpreted as type W/set of women. In the Montagovian setting relational CNs are interpreted as two-place relations (expressions of type $\langle e, \langle e, t\rangle\rangle$). Our framework allows us to treat them as dependent types, e.g. *representative* (as in *a representative of a country*) is interpreted as dependent type $c : C, r : R(c)$/for any element a in the set of countries $\|C\|$, there is a set (fiber) $\|R\|(a)$ of the representatives of that country.

Analysis Below we provide details of our compositional analysis of ILCs. The relational CN *representative* is interpreted as dependent type $c : C, r : R(c)$; the preposition *of* signals that *country* is a type on which *representative* depends; *country* is interpreted as type C. By quantifying over $c : C, r : R(c)$, we get the inverse ordering of quantifiers $\forall_{c:C}\exists_{r:R(c)}$. More specifically, the complex DP *a representative of every country* is interpreted as follows

$$\|\forall_{c:C}\exists_{r:R(c)}\| = \{X \subseteq \|\Sigma_{c:C}R(c)\| : \{a \in \|C\| : \{b \in \|R\|(a) : b \in X\}$$

$$\in \|\exists\|(\|R\|(a))\} \in \|\forall\|(\|C\|)\}.$$

Following [3] and[12], we propose to analyze Haddock descriptions as a case of inverse linking. This allows us to account for the kind of 'polyadic presupposition' that Haddock descriptions introduce. Furthermore, our proposal also accounts for the inseparability of the two quantifiers exhibited by DPs of this form (see [12], [2]) and provides a principled explanation of the locality prediction (as discussed in [3]). Crucial to the last explanation is the possibility of both 'relational-to-sortal' shifts (as in uses of relational nouns without an overt argument) and 'sortal-to-relational' ones (as in uses of sortal nouns with an overt argument).

References

[1] Jon Barwise and Robin Cooper. Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4(2):159–219, 1981.

[2] Dylan Bumford. Split-scope definites: Relative superlatives and haddock descriptions. *Ms. New York University*, 2016.

[3] Lucas Champollion and Uli Sauerland. Move and accommodate: A solution to haddock's puzzle. *Empirical issues in syntax and semantics*, 8:27–51, 2011.

[4] Justyna Grudzińska and Marek Zawadowski. Generalized quantifiers on dependent types: a system for anaphora. In *Modern Perspectives in Type-Theoretical Semantics*, pages 95–131. Springer, 2017.

[5] Justyna Grudzińska and Marek Zawadowski. Whence long-distance indefinite readings? solving chierchia's puzzle with dependent types. In *International Tbilisi Symposium on Logic, Language, and Computation*, pages 37–53. Springer, 2017.

[6] Nicholas J Haddock. *Incremental interpretation and combinatory categorial grammar*. University of Edinburgh, Department of Artificial Intelligence, 1987.

[7] Richard K Larson. Quantifying into np. *Ms. MIT*, 1985.

[8] Michael Makkai. First order logic with dependent sorts, with applications to category theory. *Preprint: http://www. math. mcgill. ca/makkai*, 1995.

[9] Per Martin-Löf. *Intuitionistic type theory*, volume 9. Bibliopolis Napoli, 1984.

[10] Robert May. *Logical Form: Its structure and derivation*, volume 12. MIT press, 1985.

[11] Andrzej Mostowski. On a generalization of quantifiers. *Fundamenta Mathematicae*, 44:12–36, 1957.

[12] Manfred Sailer. Inverse linking and telescoping as polyadic quantification. *Proceedings of Sinn und Bedeutung*, 19:535–552, 2015.

The role of Sense and Valency in the Account of Certain 'Minor' Grammatical Phenomena in Norwegian

Lars Hellan
NTNU, Trondheim

Abstract

The paper presents analyses of Light Verb Constructions and Bare Nominalizations in Norwegian from the viewpoint of articulating their properties relative to parameters of Valency and Sense. The analyses are given within a framework amenable to large scale grammars as well as to fine-grained analysis of limited phenomena, viz. the feature-based framework HPSG. While this framework has a consolidated approach to valency, this is less the case for sense, and we argue that in order for the account of sense to fit seamlessly into the overall analysis, also sense should be represented in terms of typed feature structures.

1 Formal Large Scale Grammars and 'minor' phenomena

By a large scale grammar (LSG) we may understand a grammar covering most aspects of semantics, syntax, morphology and phonology/orthography of a language, corresponding to what a reference grammar could contain. A formal grammar is a grammar where it is explicitly given whether and how the characterization of one phenomenon hangs together with the characterization of all other phenomena. Perspicuity in this respect is typically aided by the use of formalisms, supporting algorithms for the expression of complex properties and decidability of consistency. These may in turn be implemented in a parser. 'Normal' linguistics and computational grammars thereby are to be seen as a common enterprise, through the use of a formalism which serves the purposes both of linguistic analysis and of computational processing, allowing one to move consistently between fine-grained aspects of linguistic analysis on the one hand and 'consolidated' encoding of grammatical phenomena on the other.

It lies in the notion 'large scale' that nearly the totality of the phenomena of a grammar be encoded – thus, all words of the language, all syntactic patterns, all morphological patterns, both semantically compositional and semantically less compositional combinations. Among the latter belong

what scholars have over the years referred to as 'irregularity', 'minor rules', 'peripheral/non-core phenomena', 'multi-word expressions', and more. These phenomena, which constitute large portions of any grammar, are of a type that in a concise grammar require *listing* rather than generalizations, whereby they come a bit 'under the radar' of what most theoretical frameworks define themselves around, inviting for instance to the use of 'rule features'[1] and other place-holding devices. For an integrated approach like mentioned above, it is on the contrary a natural goal to replace ad hoc enumeration features by linguistically well motivated features. That is, that the phenomena be listed may be unavoidable, but the features used in distinguishing them should describe the differences accurately.

To build a formally satisfactory edifice addressing such a task, one promising avenue may be that of a uniform *feature-based* design. 'Features', broadly speaking, means 'properties' of linguistic units, where 'properties' are conceived partly relative to what is in focus of a given research, partly relative to the formal exposition of the properties. In the latter respect, 'feature structures' in formal grammars are typically attribute-value matrices (AVMs), where an attribute (the word 'feature' is here often used equivalent to 'attribute') indicates a *parameter* of specification, and a value indicates the exact value of a parameter (like 'present', for the parameter 'tense'); the 'matrix' of the AVM is constituted by a set of such attribute-value pairs, together characterizing a unit.

The framework perhaps most whole-heartedly embracing such a feature formalism is Head-Driven Phrase Structure Grammar (HPSG).[2] Here all constructs in grammar are specified in terms of *typed feature structures* (TFS), that is, a design where attributes are declared by types, and in turn take types as values, whereby specification can have any depth of recursion.[3] Types come in multiple-inheritance hierarchies, for any kind of categories or content, often of such a complexity that one needs the possibility of articulating super-sub-type relations by means of attributes indicating what sets the subtypes of a given type apart from each other, and in what respects a subtype is more specific than the super-type; in general, whenever X is a super-type of Y, this can be marked by specifications of attributes on Y for which the super-type X is not specified. Through this design, the whole grammar is in effect construed as one all-

[1] Cf. Lakoff 1967.

[2] Among essential general introductions, see Pollard and Sag 1994, Sag and Wasow 1999, Sag et al. 2003, Copestake 2002.

[3] In Copestake (2002), the following principles are given regulating feature declarations:

[A] A given type introduces the same attribute(s) no matter in which environment it is used.

[B] A given attribute is declared by one type only (but occurs with all of its subtypes).

encompassing type system, articulated into a multitude of more limited type systems introduced by attributes relative to dominating types.

The following discussion will focus on 'minor' phenomena where generalizations and analytic elegance are not so easy to obtain, but descriptive detail can nevertheless be achieved, in line with what we said above. We discuss aspects of the analysis of *Light Verb Constructions* and *Bare Nominalizations* in Norwegian,[4] both exposing a high degree of 'irregularity', and we suggest what linguistically adequate representations of these should be. This will lead to the proposal of a valency feature so far not considered in the standard literature, and to a partial proposal of how 'sense' can play a part in a formal grammatical analysis.[5]

2 Light Verb Constructions in Norwegian

2.1 Introduction

One characteristic of so-called *Light Verb Constructions* (*LVC*s)[6] is that they unfold, mostly over a sequence '*Subject* **V (P) N'**, a content that could in principle be carried by some verb ***V*** alone, and where the **N** of the sequence carries the main part of the content, hence the term 'light' for the role of the verb. The **N** thus expresses a situational content, often being 'de-verbal', and a typical role of (the 'light' verb) **V** in the LVC is to connect its *subject* to this situational content as a *role* bearer, and possibly add *aspectual* and *viewpoint* content to the situational content expressed by **N**. (1a, b, c) are examples from Norwegian; (1d, e, f) are corresponding sentences where a related verb alone carries the relational content:

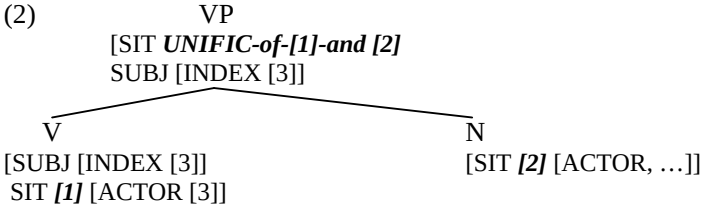
(1)
a. Han fant behag i innspillingen
He found pleasure in the recording
b. Han gjorde bruk av grammofonen
He made use of the gramophone
c. Hun stilte saken i bero
She suspended the case
d. Innspillingen behaget ham
The recording pleased him

[4] Norwegian 'bokmål'. The discussion draws on Hellan (2017), Hellan (2016, submitted), Hellan (forthcoming), and Hellan (2012, ms), and has as a background the computational grammar NorSource (Hellan and Bruland 2015).
[5] Within HPSG, our proposal has perhaps most in common with Davis 2000.
[6] Many phenomena have been characterized in terms of the notion 'light verb'; Butt 2010 may be seen for a summary of many of them. The usage of the term here employed goes back at least to Jespersen 1964, more recently Grimshaw and Mester 1988, and is a topic of much current attention, see, e.g., Piunno & Pompei 2015, Nagy et al. 2013, for European languages.

e. Han brukte grammofonen
He used the gramophone
f. Saken beror
The case is suspended

In (1b), for instance, the noun *bruk* 'use' by itself represents at least an agent and a theme/patient, and the verb *gjorde* 'made' (infinitive *gjøre* 'make, do') effects a linking of the subject *han* 'he' to the agent role. In a contrasting construction like *Den hadde vært gjenstand for uforsiktig bruk* 'It had been subject to careless use', the subject *den* is linked to the patient/theme role of *bruk*. The interplay between *bruk* and the alternating light verb expressions is thus one where the light verb is defined for linking between a certain role and the grammatical function of subject, whereas the noun has no syntactic linking, but a richer content, including the role mediated by the verb.
To give formal expression to the latter point, we can model the combination of verb and noun as in (2), representing (1b), as one where the semantic contents of these expressions 'melt together', but with a syntactic linking as 'ruled' by the light verb. The 'melting together' is indicated schematically by the content of the bracket introduced by the attribute 'SIT' (for 'Situation Structure'),[7] where the combination resides in merging, or unifying, these situational contents into one.

(2)



As indicated, the verb has ACTOR as role, as does the noun, and ACTOR being the only role carried by the verb, the situational contents can thus unify, with the role 'ACTOR' carried by the VP as a unified role. Linking to subject remains as it is encoded with the verb, since as head of the VP, the verb's valency specification is carried on to the VP.

[7] What is called SIT points towards the type of semantic space explored in traditions like Lexical Semantics and Situation Semantics (Barwise and Perry 1983); a division between such a space on the one hand and a more grammatically defined argument structure has been proposed in Melchuk (2004), and in another form in Grimshaw 2005, Levin & Rappaport Hovav (2005) and Rappaport Hovav & Leviin (1998). See Hellan (forthcoming) for an exploration of situation structure along the same lines as here. In the present analysis, this more grammatically defined level is commented on in section 3 below, represented by an attribute 'SEM' in the diagram (20) and subsequently.

A formal analysis of LVCs should partly highlight the content merger of verb and noun now indicated, and partly the manner in which the noun avails itself of the verb to have its predicative content expressed in a sentential pattern. Without making the latter a teleological issue, one may say that a situational content, in seeking linguistic expression, has more than one channel – most prominently the category of *verb*, bound to the patterns of valency offered by the grammar, but the category of *noun* is another, allowing situational content to be combined with expression of quantification, definiteness and such. In the latter case, through embedding of the noun in an LVC, the LVC provides one more possible pattern of valency for the situational content, restituting as it were the possibility of grammatical articulation of semantic roles. In this vein, one can ask: For a given SIT profile, are there principles determining which LVC pattern (of the language in question) may be used to support its realization as a *nominal* sign, and with which verbs? Languages may seem to differ as to how 'regular' they are in their use of verbs in LVCs, depending on the type of semantics of the noun. It is well recognized that LVCs as here understood constitute a major category in, e.g., Persian (cf. Karimi-Doostan 2010, 2011) and in Indic languages, and likewise in West African languages. Among European languages the profile of LVCs in Norwegian seems quite distinct from their profile in, e.g., Romance languages, and it is likely that differences at either level of comparison can be phrased in the terms suggested.

In Norwegian LVCs, verbs with 'L'-capacity – that is, being able in principle to serve as verb in an LVC - cannot freely choose among all nouns with 'N'-capacity. Moreover, a noun with N-capacity may be restricted in its LVC occurrences to only very few verbs. An interesting question is how far such instances of restrictedness follow from the meaning of the items involved – i.e., that in such cases the meaning compatibility requirement (as expressed, e.g., in the combinatorial schema above) restricts the number of possible partners. For possible cases where such reasons do not seem to obtain, we will refer to the restrictedness as *selection* – whether as the verb selecting the noun, or the noun selecting the verb. The number of LVCs in Norwegian and the proportion requiring selection, are high enough that whatever is chosen as means for formalizing the construction type, it will be pervasive in the total system.[8] In the following, we focus on these formal aspects of the analysis of LVCs, drawing partly on Hellan (2016, submitted).

[8] A proportional estimate for the Norwegian lexicon will be that it includes about 3500 de-verbal nouns. Although automatic identification of LVCs may perhaps be possible (cf. Nagy et al. 2013, for exploration relative to English), this would be a far prospect for Norwegian, especially since the number of verbs which can take part is quite high. Thus, if one wants to get a corpus-based concordance of what are possible LVCs in Norwegian, one has to start with a 'manually' established list of all verbs with L-capacity and a likewise 'manually' established list of all nouns with N-capacity, and then do string-based search for each possible V+N combination to see which ones come

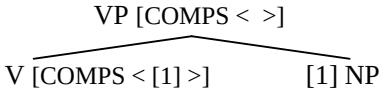
2.2 Formal analysis of LVCs

We first indicate how the constellation in (2) comes out in a formal HPSG analysis. To outline this, we first give a brief exposition of the essential combinatorial mechanism of HPSG for the verb-object constellation.

A special feature of the standard HPSG formalism is the treatment of *valence satisfaction*. Relating to a sentence like *The cat bites the dog*, we may say that *bite* has a 'program' of valence satisfaction, and such 'programs' of valence satisfaction are formalized in HPSG as *lists* of constituents for the verb to be combined with, one list for complements, standardly named *COMPS,* and one for the subject, standardly called *SPR*. For *bite*, in the use in question, these lists thus have one member each, as illustrated in (3) (a simple box '[]' inside the list bracket means one occurring item, with properties not specified):

(3)
$$\begin{bmatrix} \text{SPR} \langle \square \rangle \\ \text{COMPS} \langle \square \rangle \end{bmatrix}$$

Following a general strategy originally stemming from Categorial Grammar, valence satisfaction is registered through *deletion* of items in these lists, referred to as *cancellation.* Thus, the rule, or statement, effecting satisfaction of the valence requirements of an *object* will have schematically the form (4), where the reentrancy symbols enforce compatibility between the NP actually occurring as object, and the NP item specified in the valence list:

(4)


A similar cancellation constellation applies for subject, so that in the AVM of the top S node, both lists will be empty, this counting technically as a well-formedness requirement on the S node.[9]

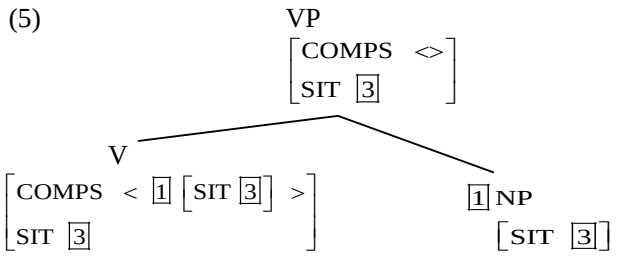
---

out (all prepositions of the language then also being taken into account). Once one has a list of possible LVCs identified by word strings, it is in turn possible to do corpus search concerning their frequency. See Hellan (2016, submitted) for further discussion.

[9] An S-node syntactic representation where both the SUBJ list and the COMPS list are empty, is of course not very informative about what is contained in the sentence. At this level, however, semantic information about the sentence's constituents will be available, and the framework also offers constructs for keeping track of the syntactic constituents., one such being 'ARG-ST', which is a list displaying the constituents in an order corresponding to their grammatical function (GF) status, without naming GFs explicitly (cf. Sag and Wasow 1999, Sag et al. 2003). Alternative formats of display more like the

Applying this formalism to the case of an LVC, the shared SIT specification between V, NP and VP is represented as in (5), the identical SIT values representing the circumstance that these specifications are unifiable (and, of course, not that they are in isolation/from the outset the same):

(5)

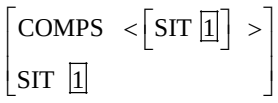


Thus, the SIT content of the verb unifies with the SIT content of the NP, this content being projected to the VP. So-called *light verbs* of course differ with regard to what their inherent content is and what constraints they may have on the object, the only point expressed in (5) is that the SIT-contents generally unify.

We then address the lexical specification of the verb and the noun. In laying out the view that the LVC so to say 'helps' combining the need for a situational content to be expressed by a noun, we assume that there are three types of nouns to take into account – *derived nominalizations* such as nouns derived with the suffix *–ing*, *bare nominalizations* (BNs) such as *løp* 'run', and *non-verbal situation nouns* ('event nouns') such as *sjokk* 'chock'. In Norwegian all three types of nouns are typical constituents of LVCs.

As anticipated in the combinations schema (5) above, when the verb is specified for taking an LVC object, the situational content of that object must be compatible with the situational content of the verb, as indicated in the schematic lexical specification (6):

(6)



If LVC combination were totally and exclusively a matter of semantic compatibility, then the combinatorial format in (5) together with the lexical specification format in (6) would be enough for regulating which combinations of verb and noun can constitute LVCs (more of course would be needed to be said about prepositional phrases in LVCs). Given the likelihood that *selection*

---

f-structure format in LFG (cf. Bresnan 2001) have also been developed (cf., for instance, Dakubu et al. 2007, Hellan and Bruland 2015).

plays an extensive role, however, we need to integrate a format for handling selection into the general format of (6).

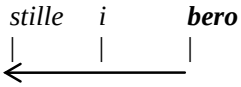
The phenomenon of selection can be first illustrated by the cases (1c,f) above, repeated:

(1c) Hun stilte saken i bero
She suspended the case
(1f) Saken beror
The case rests

The verb *bero* ('rest', as for a court case resting) has the corresponding noun *bero*, and the typical LVC for this noun is *stille i bero* ('suspend'), as in (1c). The verb *stille* 'put, position' can take many environments, being a verb of general positioning, whereas the noun *bero* has hardly other environments than the one in (1c). This means that in intuitive terms of 'selection', the noun *bero* may be seen as selecting the preposition *i* as well as the verb *stille*, as schematically indicated in (7a). This constellation contrasts with the general view of current frameworks to the effect that selection goes via the line of *government*: the preposition *i* syntactically governs *bero*, and is itself governed by the verb *stille*, as indicated in (7b), and so this would be the expected direction of selection as well:

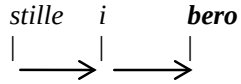
(7)

a.



'Selection' of preposition and verb by the noun

b.



Government of preposition and noun by the verb

While the construal indicated in (7b) is sustained by standard selection formalisms, a question is whether one should give formal expression also to the perspective of (7a). To lead up to that possibility, let us first spell out how the cum-government pattern of selection specification will be integrated in the schema (6). As example we use *lide nederlag* 'suffer defeat': here *lide* is among the very few 'light' verbs that can combine with *nederlag*, and vice versa. To build such a circumstance into the combination formalism, not only POS and SIT of the complement must be specified in the verb's valency frame, but also a sign-specific identification of the object noun. Schematically, this will look as

follows for the sign for *lide* (where the attribute 'KEY' serves for sign 'identity tag'; note that SIT identity now concerns a PATIENT role):

(8)
$$\begin{bmatrix} \text{ORTH} \langle \text{"}lide\text{"} \rangle \\ \text{HEAD} \left[verb \left[\text{KEY } lide\_lvc \right] \right] \\ \text{COMPS} \left\langle \begin{bmatrix} \text{HEAD } noun \left[\text{KEY } nederlag\_lvc \right] \\ \text{SIT } \boxed{1} \left[\text{PATIENT} \right] \end{bmatrix} \right\rangle \\ \text{SIT } \boxed{1} \end{bmatrix}$$

This AVM portrays the selection as exercised by a verb over the complement, i.e., *lide* selecting *nederlag*. Given that the noun equally much selects the verb, one may conceive of the selection construed the opposite way, as schematically indicated in (9), where the verb is introduced by the attribute GOVERNOR, and an expanded version of the rule schema (5) may be seen as also effecting cancellation of the list serving as the value of this attribute.

(9)
$$\begin{bmatrix} \text{ORTH} \langle \text{"}nederlag\text{"} \rangle \\ \text{HEAD} \left[verb \left[\text{KEY } nederlag\_lvc \right] \right] \\ \text{GOVERNOR} \left\langle \begin{bmatrix} \text{HEAD } verb \left[\text{KEY } lide\_lvc \right] \\ \text{SIT } \boxed{1} \left[\text{PATIENT} \right] \end{bmatrix} \right\rangle \\ \text{SIT } \boxed{1} \end{bmatrix}$$

Adding a line like this in one AVM is not very dramatic, but given that a grammar lexicon easily includes some 60-70 000 nouns, the set of attributes one generally allows in noun entries has to be calculated with some care. Stating awareness of this, we here assume 'GOVERNOR' added as a feature to the general stock.[10]

In other LVCs it may be that the verb takes nouns fairly freely, while each noun is somewhat specific in its choice of verb; such verbs are for instance *gjøre* 'do', *foreta* 'do/conduct', and *begå* 'commit', which are all agentive and combine with nouns whose content has an agentive role component. *Begå* can be used only with nouns implying that the act referred to is *bad*, like in *begå underslag* 'commit a fraud' or *begå et folkemord* 'commit a genocide'. *Foreta* rather is used with somewhat institutionalized acts, like in (10):

[10] Among features assumed inside a noun entry are foremost 'SPR' for specifiers, and COMPS for relational PPs.

(10) Banken foretok en nedskrivning av kronen
The bank conducted a devaluation of the 'krone'

Foreta will here have a lexical entry such as (11) where the specification of noun complement is not through selection, but rather through *restriction* – here that the DOMAIN of the noun must be in finance or politics (and the role ACTOR being common to verb and noun, while the noun may also have other roles, like THEME in the present case):

(11)
$$\begin{bmatrix} \text{HEAD verb } [\text{KEY } \textit{foreta\_lvc}] \\ \text{STYLE formal} \\ \text{COMPS } \left\langle \begin{bmatrix} \text{HEAD noun} \\ \text{STYLE formal} \\ \text{SIT } \boxed{1}[\text{DOMAIN } \textit{finance / politics}] \end{bmatrix} \right\rangle \\ \text{SIT } \boxed{1}[\text{ACTOR}] \end{bmatrix}$$

From the viewpoint of the noun *nedskrivning*, however, the choice of governors in an LVC is restricted to just *foreta*, hence this specification must be done through selection, i.e., mention of the GOVERNOR item in question:

(12)
$$\begin{bmatrix} \text{HEAD noun } [\text{KEY } \textit{nedskrivning\_lvc}] \\ \text{STYLE formal} \\ \text{GOVERNOR } \left\langle \begin{bmatrix} \text{HEAD verb } [\text{KEY } \textit{foreta\_lvc}] \\ \text{SIT } \boxed{1}[\text{ACTOR}] \end{bmatrix} \right\rangle \\ \text{SIT } \boxed{1}\begin{bmatrix} \text{THEME} \\ \text{DOMAIN } \textit{finance / politics} \end{bmatrix} \end{bmatrix}$$

In both cases it is the identity of the SIT specifications which marks the construction as an LVC, while the suffix '_lvc' in the KEY specification is only to keep this usage of the item in question from possible other, non-LVC usages. We assume that both *symmetric selection* as in (8)/(9) and *asymmetric selection* as in (11)/(12) are rather pervasive in Norwegian LVCs. *Symmetric restriction* may indeed be rather rare, not to speak of LVC combinations regulated by no other condition than SIT identity; but we leave attempts both at a detailed survey and possible generalizations in these terms open for the present.

Similar considerations apply for LVCs of the form 'V NP PP', like in (1c) and in (1a), the latter repeated as (13):

(13) Han fant behag i innspillingen
He found pleasure in the recording

In (13) there is a symmetric selection between *finne* and *behag*, while between *behag* and *innspillingen* there is only *restriction*, in that the NP following *i* has to indicate a STIMULUS of the experience expressed by *behag* but this could in principle be any kind of item/event. Lexical selection specifications thus have to be given for both *finne* and *behag* analogous to the above pairs, while for the governee of the preposition there are no limitations. We illustrate this constellation with the relevant entry of *finne*:

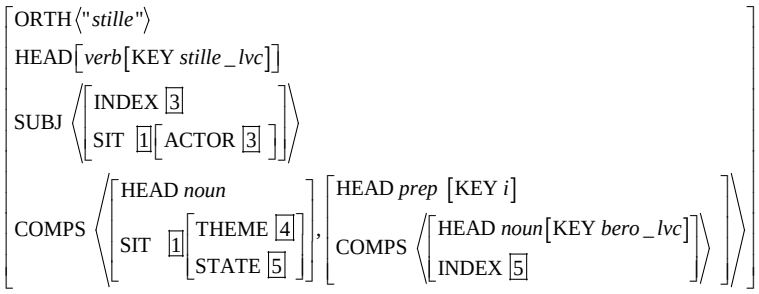
(14)
$$\begin{bmatrix} \text{ORTH} \langle \text{" } \textit{finne} \text{"} \rangle \\ \text{HEAD} \left[\textit{verb} \left[\text{KEY } \textit{finne\_lvc} \right] \right] \\ \text{SUBJ} \left\langle \begin{bmatrix} \text{INDEX } \boxed{3} \\ \text{SIT } \boxed{1} \left[\text{EXP } \boxed{3} \right] \end{bmatrix} \right\rangle \\ \text{COMPS} \left\langle \begin{bmatrix} \text{HEAD } \textit{noun } \left[\text{KEY } \textit{behag\_lvc} \right] \\ \text{SIT } \boxed{1} \left[\text{STIM } \boxed{5} \right] \end{bmatrix}, \begin{bmatrix} \text{HEAD } \textit{prep } \left[\text{KEY } \textit{i} \right] \\ \text{COMPS } \left\langle \left[\text{INDEX } \boxed{5} \right] \right\rangle \end{bmatrix} \right\rangle \end{bmatrix}$$

In (1c), however, repeated again, this is different:

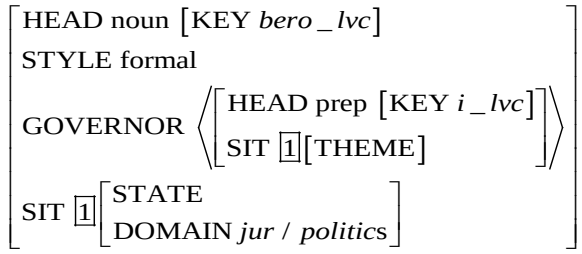
(1c) Hun stilte saken i bero
She suspended the case

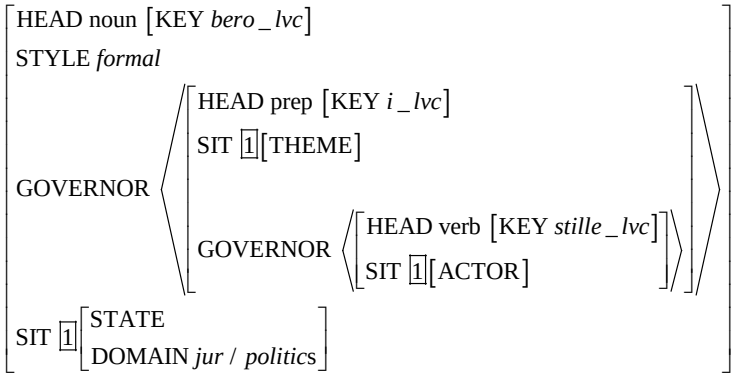
Here the selection is between *stille* and *bero*, i.e., between the verb and the governee of the *preposition*, whereas the choice of object is in principle free (although *sak* is definitely a commonly occurring noun in this frame). From the perspective of the *verb*, the specification corresponding to (14) will be as in (15):[11]

[11] In implementations of HPSG grammars like the LKB (Copestake 2002), the specification of a list within a list, as in the specification of a COMPS list within COMPS, is not licit. However, using attributes more corresponding to those used in LFG in parallel to the valency lists allows one to make specifications with the same effect as in (15). This formalism is used extensively in the Norwegian LKB grammar Norsource (cf. Hellan and Bruland 2015), but for the purposes of the present exposition we do not show this extension.

(15)


From the viewpoint of the selected noun, *bero*, however, the situation will be more complicated than for the case with *nederlag* above, since, as pictured in (7a), there is a chaining of selections, *bero* selecting *i*, and *i* selecting *stille*. As far as the relation between *bero* and *i* is concerned, it can be represented with the attribute GOVERNOR as in (16), while to cover the next selection relation up, with *stille*, it must be represented by another GOVERNOR attribute inside the first one; cf. (17):[12]

(16)


(17)


[12] The same point about specification of lists within lists holds as relative to (15).

We have thereby addressed both of the typical LVC constellations – V NP and V NP PP – both from a syntactic perspective and from a lexical perspective, in the latter respect with a view to both possible directions of selection. Throughout this, SIT identity between selector and selectee has been maintained as the formal defining feature of the construction as an LVC. We have seen that in order to fully cover all possible selection configurations, it is sufficient to introduce one relational feature, viz. GOVERNOR; however, this feature by itself is a significant addition to a set of attributes which the framework strives to keep small.[13] As for what can serve as value of SIT, i.e., sense representations, what we have shown so far are merely rather commonplace role indications. We indicate briefly in section 4 how a richer definitional basis for this domain may look.

3 Bare Nominalizations and lexical under-specification

By a *bare nominalization* (BN) we mean a noun whose form can appear as, or be similar to, the stem of a verb, and which carries no derivational affix; that is, none of the affixes standardly used for the construction of nouns from verbs. In Norwegian, such affixes include *-ing, -else, -sjon* as the most regular. For each of them, the relation to the meaning induced can vary from verb to verb, but the *gender* of the noun induced is always the same: *-else* and *-sjon* always induce masculine gender; *-ing* always induces masculine or feminine, according to norm and style.[14] BNs, in contrast, have a gender defined specifically for each noun, dependent neither on the associated verb nor on aspects of the form of the noun; some examples are given below:

Table 1. Gender of some Norwegian bare nominalizations

Masculine	Neuter
gang related to *gå* 'go'	*fall* related to *falle* 'fall'
søvn related to *sove* sleep'	*løp* related to *løpe* 'run'
	hopp related to *hoppe* 'jump'

[13] As a feature for nouns, it is related to a feature SPR (for 'specifier') used in many versions of the formalism, which encodes restrictions that a noun may exert on its determiner; still these are too different from what GOVERNOR is supposed to carry to make them instances of one and the same feature. (If one were to consider representing a so-called 'DP-analysis' of NPs, according to which the determiner is what heads an NP, with the noun as being governed by the determiner, then GOVERNOR and SPR would carry similar functions, but still of course functions different enough that they couldn't be subsumed in a single attribute.)

[14] Norwegian Nynorsk uses feminine and Norwegian Bokmål uses masculine, but with sub-norms favoring feminine.

The number of BNs in Norwegian comes close to a thousand, but we illustrate the analysis around one single instance of a *verb – BN – affix-derived noun* triplet, namely the verb *løpe* 'run', the BN *løp*, and the affix-derived noun *løping* 'running'. As noted at the outset, derivation with *-ing* leads to a gender assignment induced by the affix, such that the formation of *løping* can be conceived derivationally. The circumstance that the BN *løp* has neuter gender is a property clearly within the range of characteristic properties of this form, and characteristic properties are exactly what should *not* be supplied to an item through derivation (cf. the discussion above about 'defining only once').[15] A problem with assigning *løp* and *løping* different provenances of characteristic specification, however, is that they do have a characteristic property in common as well, namely their meaning (and of course also phonological factors). Thus, what we need is a model of grammar allowing us both to define the nouns *løp* and *løping* with a partly common semantics, and at the same time not situate *løping* and *løp* in a common derivational track from the verb *løpe*. Thus, as mentioned, a lexical entry is a structure subsumed under a general type, with certain attributes added to indicate the item specific features. In the case of Bare Nominalizations, we however face a constellation where also item-specific features have to be present in types to which lexical entries belong.

To this end we introduce a system of lexical entries functioning as a partial type inheritance hierarchy, where a sign structure with a situational semantics, but underspecified for parts of speech and valency, counting as *root*, can be inherited both by a verb and a noun entry. This design can create a noun entry with properties partly like those of a given verb, but without going via the lexical entry for that verb. For instance, the underspecified root structure may represent the concept of running, being inherited on the one hand by the verb entry *løpe* 'run', with aspectual protractedness, directional motion semantics and the corresponding valency frame, and on the other hand by the noun entry *løp*, which specifies neuter gender, but leaves valency non-determined. For visualization, the type inheritance may be portrayed as a vertical relation, as depicted in Figure 1, relative to which a track involving derivational processes could be shown as a horizontal track taking off from Verb entry. [16]

[15] Thus it will be out of the question to assume something like zero-derivational rules producing the BNs of the various genders.

[16] Designs developed in Distributed Morphology (cf., e.g., Alexiadou (2017)) have points in common with the present proposals.

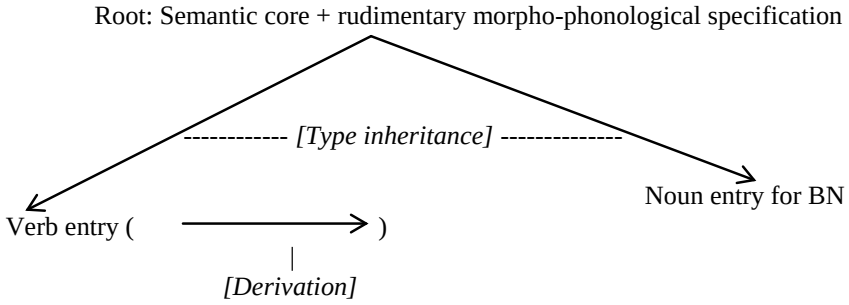


Figure 1. Inheritance design for basic lexical entries

Essential here is that while the common characteristics of the verb *løpe* and the BN *løp* are specified at the root node, their lexical entries are defined at the daughter level, not as derived constructs, but as subtypes in the type inheritance hierarchy, independently of derivational relations. Type inheritance is by necessity monotonic, so that all of the root information for *løpe/løp* is present in the lexical entry of *løp*, but beyond that the lexical entry is open for all further specification needed. For instance, from a root sign as in (18), the subsumed BN sign will be as in (19), and the subsumed verb sign as in (20); as for the latter, we assume that in addition to situational content, a verb also has a 'shallow' semantic structure SEM, representing what is often referred to as 'semantic argument structure'. In essence, SIT displays a potentially quite rich array of semantic factors, including roles and aspectual information, and reflecting situational content rather than syntactic and lexical constructs used in the expression, whereas SEM displays an enumeration of arguments under labels such as ARG1, ARG2,..., where these arguments are expressed or grammatically implicit, and the labels carry no role content per se; see section 4 for further discussion.[17]

(18) Root sign:

$$\begin{bmatrix} \text{ORTH} \langle \text{"}løp\text{"} \rangle \\ \text{SIT } run \begin{bmatrix} \text{AGENT } \boxed{3} \\ \text{MOVER } \boxed{3} \\ \text{PROTR } + \\ \text{DYN } + \end{bmatrix} \end{bmatrix}$$

[17] Cf. also Hellan 2017, and footnote 7 above. The enumeration format goes back to Tesniére 1959.

(19) BN sign:

$$\begin{bmatrix} \text{ORTH} \langle \text{"}løp\text{"} \rangle \\ \text{HEAD} \left[noun \left[\text{GENDER } neut \right] \right] \\ \\ \text{SIT } run \begin{bmatrix} \text{AGENT } \boxed{3} \\ \text{MOVER } \boxed{3} \\ \text{PROTR } + \\ \text{DYN } + \end{bmatrix} \end{bmatrix}$$

(20) Verb sign:

$$\begin{bmatrix} \text{ORTH} \langle \text{"}løpe\text{"} \rangle \\ \text{HEAD} [verb] \\ \text{SUBJ} \left\langle \left[\text{INDX.} \boxed{1} \right] \right\rangle \\ \text{SEM} \begin{bmatrix} \text{PRED } løpe\_rel \\ \text{ARG1 } \boxed{1} \end{bmatrix} \\ \\ \text{SIT } run \begin{bmatrix} \text{AGENT } \boxed{1} \\ \text{MOVER } \boxed{1} \\ \text{PROTR } + \\ \text{DYN } + \end{bmatrix} \end{bmatrix}$$

We also illustrate nominalization derivation with *-ing* from the verb structure (20). We assume that nouns do not take syntactic arguments, beyond those that are generally possible with relational nouns, but that in addition to the situational content, also the 'shallow' semantic structure SEM is retained in the derived item.[18]

[18] We are of course not assuming that derivation is generally "productive" (cf. Chomsky 1972)) – a nominalizing affix may apply to only a few verb stems; but as long as the resulting properties can be packed into a many-instance package rather than be assigned anew for each case, the economy gain of derivation is obtained.

(21) Sign for the noun *løping* 'running':

$$\begin{bmatrix} \text{ORTH} \langle \text{"}løping\text{"} \rangle \\ \text{HEAD} [noun] \\ \text{SEM} \begin{bmatrix} \text{PRED } løpe\_rel \\ \text{ARG1 } \boxed{1} \end{bmatrix} \\ \text{SIT } run \begin{bmatrix} \text{AGENT } \boxed{1} \\ \text{MOVER } \boxed{1} \\ \text{PROTR } + \\ \text{DYN } + \end{bmatrix} \end{bmatrix}$$

Thus, both (19) and (20) are pure expansions from (18), observing monotonicity of feature inheritance, while in the derivational process from (20) to (21), although essential information is carried on, the mapping is not monotonic.

This analysis instantiates the situation mentioned initially, i.e., that while a lexical entry is always a structure subsumed under a general type, with certain attributes added to indicate the item specific features, in the case of Bare Nominalizations, also item-specific features can be present in types subsuming lexical entries. This is formally fully feasible, but it will be interesting to see how such a design will scale. Technically, perhaps the most immediate point to be developed is what to enter in the ORTH attribute, and the corresponding attribute for phonological specification.

We may mention one immediate consideration relative to the issue of specification of LVC selection discussed in the previous section. For a BN, like *nederlag* in (9) or *bero* in (17), the value of GOVERNOR will be specified in the expanded BN entry, with the choice of light verb counting as a characteristic property of the noun. The same goes for event nouns, where Root and BN entry are identical. But how about derived nouns – the model will suggest that characteristic properties of, e.g., *-ing*-nouns, are not specified in their noun entry, only in the entry of the verb and in the specification of the derivational affix. Choices made by the noun between such general verbs as *foreta, gjøre, undergå*, etc., must then be predictable from compatibility between the role inventory of the noun and that of the verb, and compatibility between aspectual features of the noun and those of the light verb. That is, for derived nouns, the use of the attribute GOVERNOR should be in principle superfluous. While intuitive estimates do not go against this position, it is a matter to be pursued in a separate setting.

4 Senses

A key point developed in section 2 is that semantic specifications of combining items can *unify*, this being a key characteristic of light verb constructions. An essential point made in section 3 is that the semantic specification is part of the common denominator between a verb and a noun represented in an

underspecified lexical sign. In both cases this semantic specification reflects sense, not argument structure, and we have formalized it as an AVM introduced by the attribute SIT, for 'situation', with standard role labels figuring as attributes. We now look more closely at the organization of this domain of specification.

In accordance with the general design of typed feature structures, attributes are declared by types. Below is an excerpt from a type hierarchy where under a type node we indicate which attributes (if any) are introduced by the type. The hierarchy combines Aktionsart (cf. Vendler 1967, Smith 1997) with standard situational notions. Note that the English verb names under the leaf nodes are mnemonically convenient names of *types* as defined in terms of the inheritances shown, not of the English *verbs* so named.

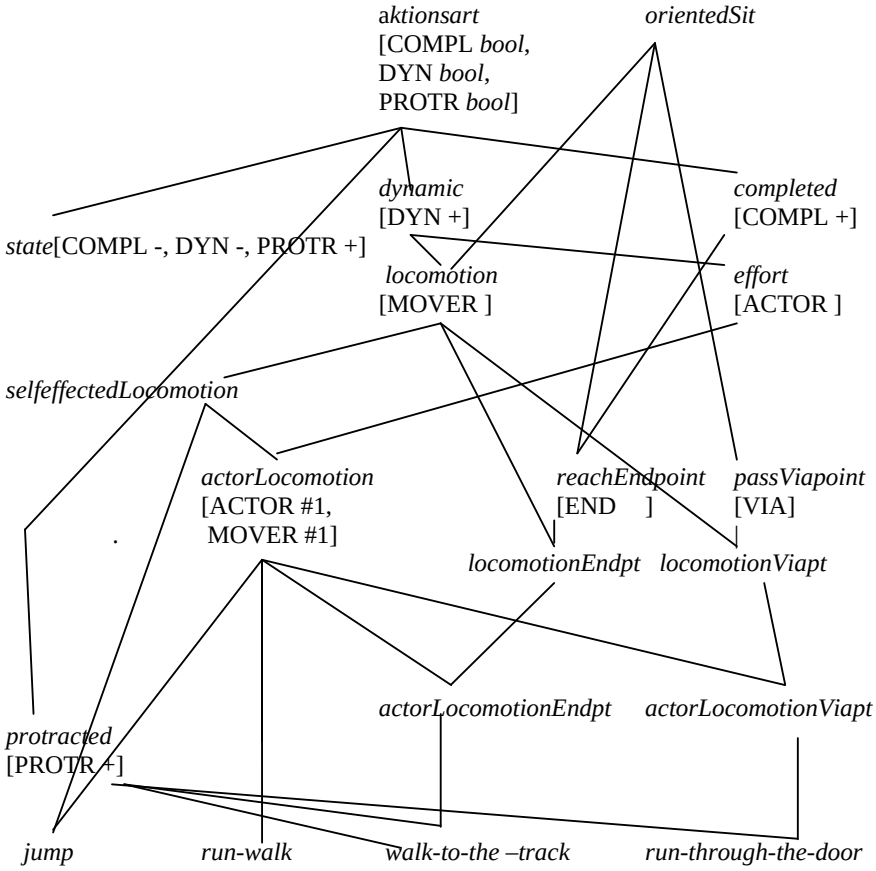


Figure 2. Partial type system integrating situation types and *Aktionsarten*

.

Once introduced, an attribute is inherited to all subtypes of the type, for instance as indicated in (22) for the type *actorLocomotionViapt*, suitable, for instance, for 'run-through-the-door':

(22)
$$\textit{actorLocomotionViapt}\begin{bmatrix}\text{ACTOR } \boxed{1}\\ \text{MOVER } \boxed{1}\\ \text{VIA}\\ \text{DYN } +\\ \text{PROTR } +\end{bmatrix}$$

Clearly, however, if one aspires to represent senses, the mere display of roles and Aktionsart is not enough. For instance, for distinguishing between *locomotion verbs* such as 'run', 'walk', 'climb', etc., as indicated in Table 2 below,[19] relevant factors include kind of entity, dimension of incremental process, pattern in which iteration happens, if any, velocity, medium in which the action takes place (water, air, ground ('terra ferma', abbr. 'terfrm')).

Table 2 Distinguishing criteria for Locomotion verbs of iteration

German	English	Norw	KIND	INCREM-DIM	ITER-PATT	VELO CITY	PART/ ORGA N	MEDI UM
Laufen	*run/go*	*løpe/gå*	legged-anim	horisontal	zip-lock	unmark ed	all-legs	terfrm
Gehen	*walk*	*gå*	legged-anim	horisontal	zip-lock	unmark ed	all-legs	terfrm
Rennen	*run*	*løpe*	legged-anim	horisontal	zip-lock	fast	all-legs	terfrm
Schlendern	*stroll*	*slentre*	human	horisontal	zip-lock	slow	both-legs	terfrm
Spazieren	*stroll*	*spasere*	human	horisontal	zip-lock	slow	both-legs	terfrm
Klettern	*climb*	*klatre*	legged-anim	vertical	zip-lock	unmark ed	all-limbs	terfrm
Hinken		*hinke*	2- egged-anim	horisontal	success ion	unmark ed	one-leg	terfrm
	creep	*krype*	Anim	horisontal	success ion	unmark ed	under-side	terfrm
Fliegen	*fly*	*fly*		all	success ion	unmark ed	wings/ arms	air
Schwimmen	*swim*	*svømme*	All	all	succesi on	unmark ed	All-limbs	water

[19] Based on unpublished work by Dorothee Beermann.

In terms of AVMs, the corresponding display of distinguishing properties for a verb like 'run' can be stated as in (23) (cf. Hellan (forthcoming) for details[20]):

(23)
$$\textit{run}\left[\begin{array}{l}\text{ACTOR } \boxed{1} \\ \text{MOVER } \boxed{1}\ \textit{running-partcpnt}\left[\begin{array}{l}\text{ITERATN } \textit{zip-lock} \\ \text{INCREM-DIM } \textit{horisontal} \\ \text{PARTorORGAN } \textit{leg} \\ \text{VELOCITY } \textit{fast} \\ \text{MEDIUM } \textit{terfrma}\end{array}\right] \\ \text{DYN } + \\ \text{PROTR } +\end{array}\right]$$

Thus, the TFS formalism allows one to give semantic descriptions down to the minutest detail, and no matter the degree of detail, operations of unification apply in the same way.

5 Final remarks

We have shown how a level of rather detailed semantic analysis can be applied in the account of certain phenomena which by themselves are of the type that in a grammar require listing rather than generalizations. What the present approach offers is not to avoid listing, but to distinguish between the listed items through articulate description rather than 'rule features' or other place-holding devices.

Relative to a large scale grammar, such specification will come at a cost, for one thing relative to processing if built into/as a parser, for another relative to the work of developing adequate analyses of the very many instances of the phenomena. If one wants the system to actually work as a parser or as an easily navigable database, one will want to be able to maintain it in a 'consolidated' version. For the light verb constructions, for instance, a likely strategy for a large scale grammar is to ignore their 'light verb' status entirely, and just treat them like any other transitive constructions. For the Bare Nominalizations, the corresponding strategy will be to just treat them like any other nouns, perhaps marked as 'relational' if a distinction relational – non-relational can be maintained for a realistic inventory of nouns (at least 50,000, probably). More in-between strategies might try to deploy the 'SEM' level of semantic representation but leave out the SIT level, a move which would

[20] The value of 'MOVER', viz. *running-partcpnt*, here introduces a new type hierarchy for qualification of entities as participants, whereby the array of attributes and values is organized formally in the same way as the types and attributes indicated in Figure 2.

comply with current designs of analysis in many domains,[21] although for the phenomena here considered, the sense factors involved are clearly not bound to semantic argument structure.

It's important to note that such a recourse to the 'consolidated' is not to be seen as a sacrifice of insight to the purpose of efficient processing or navigation: in the setting we have defined, the inter-weaving of semantic detail in formal and lexical structure is a fact which a grammar has to account for, since this is actually part of the total symbolic system. But in order to be able to represent the interplay, the grammarian must be able to define both the semantic detail per se, and the 'rest' grammar as a whole. Whether one attains the right modularization is always a question, which however cannot be resolved without making concrete attempts.

Such 'attempts' are of course of an order of effort requiring 5-10 years to represent reliable coverage of the language, and so there are additional concerns how they can be sustained in a consistent fashion. Being computational, as mentioned at the outset, is perhaps a prerequisite, but this also is of little avail if the system is not shared between many people, transparently enough to be discussed. A further factor enhancing consistency is if the system serves in an actual application, whose maintenance is dependent on consequential consistency.

From a linguistic viewpoint, this nevertheless all has to come down to a system which makes sense both in the large and in the minor perspectives, since grammar, as a factual symbolic entity, comprises all of the aspects mentioned. The formalism we have discussed seems to be a possible vessel for such an integrated design.

As a final remark, it may be noted that 'meaning' is not necessarily something we detect only when looking into lexical semantics, construction types and quantification. The fact that any speaker easily navigates/masters lexical systems with 10-15,000 verbs, 80-100,000 nouns, and 10-15,000 adjectives, suggets the possibility that this mastery does not just reside in a large memory of pairings of words with their individual meanings, but that there are regularities of organization of form and meaning, for instance regarding valency frame inventories, and distribution of valency frames and valency frame clusters, across the verbs of a language.[22] Possible regularities in such domains cannot be studied unless one already has well structured, if not exactly fine-grained, descriptions of large inventories of verbs, preferably all of the 10-15,000 verbs. Here, thus, large scale grammar may once again offer helpful input for linguistic investigation.

[21] It for instance represents the approach to the display of logical structure used in Minimal Recursion Semantics (cf. Copestake et al. (2005), the HPSG counterpart of Montagovian semantics (Montague 1974).

[22] Cf. The initial discussion of 'valency pods' and 'valency classes' in Hellan et al. (2017), Hellan (to appear), Dakubu and Hellan (submitted).

References

Alexiadou, Artemis. 2017. On the complex relationship between deverbal compounds and argument supporting nominals. In: A. Malicka-Kleparska & M. Bloch-Trojnar (eds) *Aspect and Valency in nominals*. Berlin: Mouton de Gruyter.

Barwise, John and John Perry. 1983. *Situations and Attitudes*. MIT Press.

Bresnan, Joan. 2001. *Lexical Functional Syntax.* Blackwell.

Butt, Miriam. 2010. "The Light Verb Jungle: Still Hacking Away". In: *Complex Predicates in Cross-Linguistic Perspective.* Ed.by M. Amberber, M. Harvey, and B. Baker. Cambridge: Cambridge University P ress, pp.48–78.

Chomsky, Noam. (1957) *Syntactic Structures*. Mouton, The Hague.

Chomsky, Noam. 1970/1972. Remarks on Nominalization. In Roderick Jacobs & Peter Rosenbaum (eds.), *Readings in English Transformational Grammar*, 184–221. Waltham, MA: Ginn.

Copestake, Ann. 2002. *Implementing Typed Feature- Structure Grammars*. Stanford: CSLI Publications.

Copestake, Ann, Dan Flickinger, Ivan Sag and Carl Pollard. 2005. Minimal Recursion Semantics: An Introduction. *Journal of Research on Language and Computation*. 281-332.

Dakubu, Mary Esther Kropp, and Hellan, Lars (submitted). Verb Classes and Valency Classes in Ga. Presented at SyWAL II (Symposium on West African Languages), Vienna, 2016.

Davis, Anthony. 2000. *The Hierarchical Lexicon*. Stanford; CSLI Publications.

Gholamhossein Karimi-Doostan. 2010. Lexical categories in Persian. *Lingua*. 1719. 1-14,

Gholamhossein Karimi-Doostan. 2011. Separability of light verb constructions in Persian. *Studia Linguistica*. 1–26.

Grefenstette, G. and S. Teufel 1995. "Corpus-based method for automatic identification of support verbs for nominalization". In *Proceedings of the 7th Meeting of the European Chapter of the Association for Computational Linguistics* (EACL'95).

Grimshaw, Jane, and Armin Mester. (1988). Light Verbs and Θ-Marking. *Linguistic Inquiry* 19(2):205–232.

Grimshaw, Jane. 2005[1993]. Semantic structure and semantic content in lexical representation. In *Words and Structure*, Jane Grimshaw, 75–89. Stanford CA: CSLI.

Hellan, Lars. 2016. Light Verb Constructions as valency modeling. A study of Norwegian. Presented at SLE 2016, Naples. Submitted for proceedings as: Unification and selection in Light Verb Constructions. A study of Norwegian.

Hellan, Lars.. 2017. A design for the analysis of bare nominalizations in Norwegian. In: A. Malicka-Kleparska & M. Bloch-Trojnar (eds) *Aspect and Valency in nominals*. Berlin: Mouton de Gruyter.

Hellan, Lars. Forthcoming. Situations in Grammar. In Bodomo, A., Essegbey, J. and Kallulli, D. (eds). Title and publisher to be announced.

Hellan, Lars. To appear. Grammar as a symbolic system. (Editor, title and publisher to be announced.)

Hellan, Lars, Malchukov, Andrej; Cennamo, Michela. 2017. Introduction: Issues in contrastive valency studies. In Hellan, Lars, Malchukov, Andrej; Cennamo, Michela. eds. 2017. *Contrastive Studies in Verbal Valency.* John Benjamins Publ. Co., Amsterdam, 2017.

Hellan, Lars and Tore Bruland. 2015. A cluster of applications around a Deep Grammar. In: Vetulani et al. (eds) *Proceedings from The Language & Technology Conference* (LTC) 2015, Poznan.

Jespersen, Otto. 1965. *A Modern English Grammar on Historical Principles*, Part VI, Morphology. London: George Allen and Unwin Ltd.

Lakoff, G. (1970) *Irregularity in Syntax*. Holt, Rinehart, and Winston.

Langacker, Ronald. 2008. *Cognitive Grammar: A basic introduction*. New York: Oxford University Press.

Levin, Beth & Hovav, M. (2005). *Argument realization*. Cambridge: Cambridge University Press.

Melchuk, Igor. 2004. Actants in semantics and syntax I: actants in semantics. *Linguistics* 42-1: 1-66.

Nagy, István, T., Veronika Vincze, and Richárd Farkas. (2013) Full-coverage Identification of English Light Verb Constructions. *International Joint Conference on Natural Language Processing*, pages 329–337, Nagoya, Japan, 14-18 October 2013.

Pollard, Carl and Ivan Sag (1994) *Head-driven Phrase Structure Grammar*. Chicago Univ. Press.

Pompei, A. and V. Piunno (2015). Light Verb Constructions. An interlinguistic analysis to explain systemic irregularities. Paper presented at SLE 2015, Leiden.

Rappaport Hovav, Malka & Levin, Beth. 1998. Building verb meaning. In *The Projection of Arguments*, Miriam Butt & Wilhem Geuder (eds), 97–134. Stanford CA: CSLI.

Sag, Ivan, and Tom Wasow. 1999. *Syntactic Theory. A formal introduction*. Stanford: CSLI Publications.

Sag, Ivan, Tom Wasow and Emily Bender. 2003. *Syntactic Theory. A formal introduction*. Stanford: CSLI Publications.

Samardzíc, Tanja and Paola Merlo. 2010. Cross-lingual variation of light verb constructions: Using parallel corpora and automatic alignment for linguistic research. In *Proceedings of the 2010 Workshop on NLP and Linguistics: Finding the Common Ground*, pages 52–60, Uppsala, Sweden, July. ACL.

Smith, Carlota. 1991, 1997. *The parameter of aspect*. Dordrecht: Kluwer.

Tesnière, Lucien. 1959. *Éleménts de syntaxe structurale*. Paris: Klincksieck.

Vendler, Zeno. 1967. *Linguistics in Philosophy*. Ithaca, NY: Cornell Univ. Press.

Wierzbicka, Anna. 1996. *Semantics, primes and universals*. Oxford: Oxford University Press.

Planning for Natural Language Generation in GF

Gleb Lobanov*[1] and Krasimir Angelov†[2]

[1]Immanuel Kant Baltic Federal University
[2]Chalmers University and University of Gothenburg

Abstract

In this paper we present a minimalistic approach to document planning in natural language generation which is specifically targeted to the Grammatical Framework (GF) system but it also generalizes to other formalisms with a type theoretical abstract syntax. The advantage of type theory is that it already has in place enough mechanisms to constrain the set of possible alternatives for planning the document. It turns out that document planning is easy to describe as a type theoretical proof search under a linear context. We present the method and demonstrate it within the use case of weather report generation.

1 Introduction

Natural language generation (NLG) is a ubiquitous problem with many possible applications. The usual scenario is that we have some kind of machine readable data source from which information is extracted and then used for the generation of text in a natural language. The step from the raw data to a coherent text, however, is not straightforward. It requires planning of the overall structure of the text (document planning), then micro planning of low-level details, such as word choice, and, finally, the rendering of the text with all of its intricacies with respect of the grammaticality (linearisation, surface realisation). The problem is further complicated if simultaneous generation in several languages at once is required.

In this paper we will focus on the document planning in natural language generation as it is now implemented in Grammatical Framework (GF) [16]. GF is a formalism where grammars for one or more natural languages can be coupled together via an abstract syntax. The abstract syntax is a logical framework [9] where the grammarian can describe the possible syntactic and semantic constructions of the target domain in a language independent

*globanov@kantiana.ru
†krasimir@chalmers.se

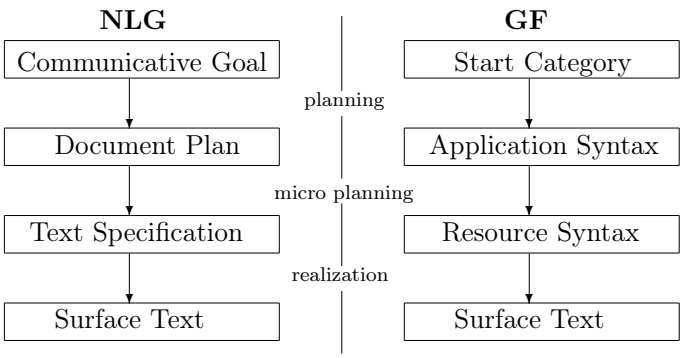


Figure 1: Natural Language Generation steps in standard terminology and in the GF terminology.

way. In this way the abstract syntax plays the role of an interlingua. The representation of a sentence in that interlingua is a lambda term which must be well-typed with respect to the logical framework. Given a lambda term it is always possible to render it in multiple languages at once given that there is a grammar for each concrete language.

In the next section we will describe how the framework relates with the usual NLG concepts.

1.1 NLG Pipeline

In the NLG parlance [17] there are three main steps in the generation of text (see Figure 1): document planning, micro planning, and surface realization. The inputs to the first step are the given communicative goal, i.e. what kind of document we want to generate, and raw data of some kind, which contains the information that needs to be communicated. As part of the planning, the raw data is converted to a set of messages that are about to be included in the document. The messages do not necessary correspond directly to the raw data. For instance we might decide that parts of the raw data are uninteresting and can be omitted. Similarly some of the messages might be computed algorithmically from the data, e.g. by reinterpreting the input in different metric systems, or by synthesizing new information which represents trends in the input. For example, in the weather report we compare the current forecast with the typical average weather for the season, which generates interesting messages that are not in the original data.

In the second step – the micro planning – the overall plan is refined by making the lexical choice for the individual words, deciding how to aggregate messages into bigger phrases, and even planning for potential anaphoric expressions in order to avoid repetitions. The result is the text specification which is a data structure describing the text which is about to be generated.

The last step is to render the text specification into an actual text. In this step we can also produce formatting instructions for a markup such as HTML or LaTeX depending on the application.

The steps of an NLG pipeline are shown on Figure 1, together with the correspondences in GF. The framework offers concepts and processes for everything but for the document planning. This reflects the fact that, so far, when GF was used for NLG, the input was usually a structure which can be rather strightforwadly converted to a lambda term which is then the input for micro planning. Examples for that can be found in [8], [10] and [5], where the input is either a mathematical formula or a type-theoretical proof which is then rendered as text.

A notable exception is Dana Dannélls museum project [6] where the input is database records. She uses templates and provide them with particular constituents, the selection of which is explicitly controlled. Contrarily, we delegate this task to the existing and modified GF proof-search mechanism. It saves from labor intensive description of conditional rules for template population, which could became exhausting when templates are comprehensive and numerous. For the same reason, it also makes available variability of the output text at no cost because several proof objects could be found for one and the same target type.

Now we should first review the GF concepts from Figure 1, and after that we will proceed with the document planning phase that we have developed together with the use case for weather reports. Characteristic to weather reports is that they are described as a set of disconnected facts, such as temperature and humidity, which have to be joined together by the document planner into a tree structure that describes the desired document structure. The later is then send for micro planning.

1.1.1 Start Category

In the usual NLG parlance, document planning starts by choosing a communicative goal, while the equivalent in GF is to choose a start category. The concept comes from formal grammars where one of the syntactic categories is designated as the category from which parsing with the grammar should start. Since GF is also a logical framework, we can also do proof search, in which case the start category corresponds to the type (proposition) for which we want a proof. As we will see later, document planning is a special case of proof search and whence the communicative goal is nothing else but a specific category.

1.1.2 Resource Syntax

We should clarify that the framework distinguishes two main types of grammars: resource grammars and application grammars. Both kinds of grammars are developed in the GF programming language but they are designed for very different purposes.

The resource grammars are part of the Resource Grammars Library [15]

which is a collection of wide coverage linguistic grammars that all share a common abstract syntax. This syntax tries to abstract as much as possible from the language specific details but usually goes only as far as syntactic transformation. Other phenomena like multiword expressions, idioms, pragmatics, etc. are not covered since they are highly language specific. In addition, the resource grammars might include language specific extensions for cases where the common interlingual Application Programming Interface (API) is insufficient. What the resource grammars are good for is to be used as libraries for building applications. The use of the library simplifies the development of the application by freeing the developer from low-level details like word order and gender agreement. Furthermore, although the resource library does not hide all language specifics, it might still be quite good in capturing language independent syntax for most cases. Usually when an application has to cover multiple languages at once then we start by implementing the first language with the library, and then for the second language we just modify the parts that need changes. In practice we find that thanks to the shared abstract syntax of the library this process is a lot more efficient than starting the second language from scratch.

The resource grammars are very linguistic in nature; the abstract syntax there can be seen as a linguistic ontology of the possible syntactic constructions that one can expect to find in any language. Each concrete syntax then defines how these constructions are realized in that language. For example, the abstract syntax for a phrase like:

It is warm. The temperature is 20°C but it feels like 25°C.

is a complicated tree structure like:

```
(UseCl (TTAnt TPres ASimul) PPos
       (ImpersCl (UseComp (CompAP <tempType>))))
(SSubjS (UseCl (TTAnt TPres ASimul) PPos
               (PredVP (DetCN (DetQuant DefArt NumSg)
                              (UseN temperature_N))
                       (UseComp (CompNP <temp>))))
        but_Subj
        (UseCl (TTAnt TPres ASimul) PPos
               (PredVP (UsePron it_Pron)
                       (ComplSlash (SlashV2a feel_like_V2)
                                   <appTemp>)))))
```

where the slots `<tempType>`, `<temp>` and `<appTemp>` has to be filled in with actual values such as `warm`, 20°C and 25°C. Here every function encodes the use of a specific construction. For instance `UseComp`, encodes the use of the copula verb, which in English is one of the forms of the verb "be" while in Russian the same can be rendered by just omitting the verb altogether.

The resource syntax is no doubt quite complicated but it has to be because it encodes the surface text down to every detail. This is in fact what corresponds to the text specification in the NLG terminology.

1.1.3 Application Syntax

The application grammars are targeted to a particular domain and this makes it possible to take care of all pragmatic issues that the resource grammars have ignored. We still utilize the resource grammars as libraries, i.e. instead of defining each specific grammar on the level of strings, we use functions that are already defined in the libraries. In fact it is recommended to work on even higher level by using an API which wraps commonly used combinations in overloaded macroses which simplify the encoding. For example for every construction `C` there is a macro called `mkC` which creates that construction with a different set of arguments. This makes the API both more compact and easier to memorize.

Contrary to the resource grammars, the application grammars have a very domain specific abstract syntax. For example the above deep and complicated tree is reduced to just:

```
InfoTemperature <tempType> <temp> <appTemp>
```

where we still have the same slots for the actual feature values. The job of the application grammar is to define the mapping.

In NLG the application syntax corresponds to the document plan and thus the process of micro planning corresponds to application grammar writing. The difference is that while in other NLG systems the micro planning happens dynamically during the language generation, here it happens statically during the compilation of the grammar. Furthermore, since all resource grammar functions get in-lined in the definition of the application grammar, the final grammar is able to do simultaneously micro-planning and rendering with no run-time overhead.

Note that the application grammar has a different concrete syntax module for each language in the domain. This means that the micro planning can be done differently for each language. This is exactly what makes the final applications multilingual: all steps prior to the micro planning, e.g. document planning, are typically language independent.

2 The Weather Report Case

As an illustration for document planning we have chosen the weather report case, which is a practical application involving uncomplicated analysis of the data at hand. It exhibits grammar-construction principles without distractions caused by peculiarities of more uncommon domains with hard-to-comprehend special languages. The application receives data in a JSON format from the weather forecasting cloud service DarkSky[1], and builds a document plan.

[1] http://www.darksky.net

2.1 Abstract syntax

The abstract syntax of the weather report grammar is modular, reflecting the compositional structure of the text. It comprises three modules: atomic messages, composite messages, and rhetorical structures. Atomic messages are functions with one or no arguments, which stand for one value or for an abstract concept from the raw data source. The type of the value is the target type of the corresponding function. Composite messages represent phrases or sentences in the text and are functions with several arguments which are applied to either atomic messages or other composite messages. The target types of the composite messages depend on their roles in the rhetorical structures. The later are defined in the third module, which consists of functions bringing together text spans and producing the schemata of the Rhetorical Structure Theory [11].

2.2 Atomic messages

As we said atomic messages stand for values from the raw data source. They could also represent abstract concepts, qualities or other bits of information which aren't supported by values, but are necessary constituents of a discourse. For example, the following are two of the atomic functions which correspond to values from Table 1, which shows the main data values used in the weather report production:

```
TemperatureVal : Float -> Temperature
ExtremelyHot : TempType
```

Here, function `TemperatureVal` takes a real number and returns a value-carring message of type `Temperature`. In contrast, function `ExtremelyHot` doesn't require any arguments and has target type `TempType` denoting human perception of temperature levels.

2.3 Composite messages

When linearised, composite messages are phrases or sentences. They consist of atomic messages and we use them to deliver a complete utterance about some state of affairs, action or process. In the abstract syntax they are formalized as functions with one or more arguments, where the types of the arguments must be target types of functions for atomic messages. Target types of composite messages are either `Nucleus` or `Satelite` – concepts from Rhetorical Structure Theory, as will be discussed later. An example of a composite message is `InfoPrecipType`:

```
InfoPrecipType : PrecipIntensity -> PrecipType -> Satellite
```

It takes an atomic message `PrecipIntensity`, which is a short description of precipitation intensity together with its value (millimeters of water),

and an atomic message `PrecipType`, which stands for a precipitation type, e.g. snow, rain, etc. The target type `Satellite` signifies the role of the composite message which it plays in more complex rhetorical structures defined in the abstract syntax.

2.4 Rhetorical Structures

The next structural level after the messages is the rhetorical structure. Rhetorical Structure Theory (RST) [11] describes text structure in terms of relations between its constituents. Each relation holds between a core element, a nucleus, and a set of supporting elements, satellites. A theory exhibits a comprehensive list of schemata describing different combinations of relations, nuclei, satellites, and constraints on them. Below, there is an example of a schema formalisation, which has `Background` as a core relation:

```
cat Nucleus
    Satellite
    SatelliteList
    Schema

fun
  BSat : Satellite -> SatelliteList
  CSat : Satellite -> SatelliteList -> SatelliteList

  Background : Nucleus -> SatelliteList -> Schema
```

We have settled on RST because it doesn't depend on any particular language and text type or purpose. This feature is beneficial because the weather report case produces weather summaries in English and Russian. Besides, it has a wide recognition in an NLG community [7]. Alternatively, [14] outlines and formalises a document structure using a type theory with dependent types. His results combined with RST could be appropriate and practical for our approach, taking into account that GF is essentially a logical framework — a lambda calculus with dependent types. However, for the weather report case we confined ourselves with some basic RST concepts.

2.5 Robust Generation

A well structured grammar should be able to generate reasonable text for any possible input comming from the data source. In the simplest case the input is a set of values for features such as temperature, precipitation, etc. However, it can happen at any time that each of the values is missing. For example there might be no information for the temperature, or perhaps the precipitation intensity is unknown. Despite this the grammar must be robust, i.e. its abstract syntax must contain at least one tree for every

possible set of available features. This is trivially satisfied if the abstract syntax contains a list like structure which presents the atomic messages one by one, perhaps in different sentences.

The trivial solution, however, is unsatisfactory since it will just produce a list-of-values like output disguised as a text. In order to avoid that we usually need a more refined rhetorical structure which can aggregate several atomic messages into one composite message. The function `InfoPrecipType` from the previous section is an example of this. The introduction of composite messages makes the generated text more fluent but makes the grammar less robust. Now if either the precipitation intensity or the precipitation type is missing then we cannot communicate the other because the composite message demands both of them.

There are two possible strategies to ensure that the generation is robust. The first is to keep the grammar untouched but to massage the set of atomic messages. For example if we do not have information about the precipitation intensity then perhaps there is no precipitation at all so it is better to say nothing at all than to generate clumsy statements like "there will be snow of unknown intensity". In order to achieve this, during the initial transformation of the input data to a list of messages, we can check for the intensity and if it is missing then we can also erase the information for the intensity type. In other words we filter out uninteresting values from the data source and no corresponding messages are generated.

The other solution is to alter the grammar. For the same example from the above this would mean that we have to add two more messages – one that requires only `PrecipIntensity` and another with only `PrecipType` as an argument. In this way even if one of the values is missing we could still verbalize the other.

Which of the two solutions is appropriate is a design question. In general to ensure that a grammar is a robust verbalizer we must first segment the set of input values into dependency groups. The characteristic of a dependency group is that whenever one of the values in the group is present, the other values are present as well and vice verse. A prototypical example from our domain are the current: year, date, month, day of the week, etc. They are all derived from the value of the date for which we must produce a weather report, so they are always simultaneously available. In other cases, like for `PrecipIntensity` and `PrecipType` we decided to put them in the same dependency group by filtering out the original information from the data source.

Once we know the set of dependencies, then a necessary condition for the grammar is that it must contain at least one message for every dependency group which takes as arguments all and only the values from that group. Other composite messages that combine values from different dependency groups could still be used for aggregation. The rhetorical structure of the grammar must also allow for these messages to be included in the document.

3 Document Planning

Document planning is a search problem where we must find an abstract expression which includes all and only the necessary messages. In order to automatize the problem, we have implemented a new document planner which is a variant of the regular proof search methods in GF.

In the existing API [2] there are already methods where given a target type, the framework can search for either all or a random subset of the abstract expressions of that type. Since the framework is an implementation of Martin Löf's type theory [12], the algorithms already implement all constraints that can be enforced by using dependent types. In addition, the order in which expressions are selected is controlled with a statistical model [3] – for exhaustive generation the result is a sequence of expressions in decreasing probability order, and in random generation the chance of an expression to be selected is determined by its probability. Both APIs are extensively used in applications. For example it is common to generate random expressions in order to check if a grammar generates valid linearisations in all languages. Exhaustive generation on the other hand can be used to obtain a sample of the space of all possible expressions. For example in [1] it has been used to combine keyword search with grammar based search.

The search strategy in GF is inspired by the choice sequence interpretation of type theoretical terms [13]. We start with an expression which is just a meta variable of the initial type. After that on every step we substitute the left most meta variable with a function which is applied to enough fresh meta variables to saturate its type. The type of the function application should also unify with the type of the meta variable that it substitutes. In this way every complete expression is just the final state in a sequence of partial expressions. Moreover, on every step there could be several alternative choices to be made for the current meta variable. The set of all possible states then naturally forms a choice tree.

Figure 2 shows a toy abstract syntax and its corresponding choice tree. This example does not include dependent types and moreover all functions have only first order types. High-order functions can take as arguments lambda abstractions which are useful for instance for modeling anaphoric references [16]. Similarly dependent types can express complex semantic constraints. Both features are useful but we take them for given from the existing search algorithms that the framework already provides. Their presence or absence does not add anything new to the design of the planner. For that reason the simplified example on the figure suffices to illustrate our design.

In the example the search starts with a meta variable of the designated initial type S. In the first step we replace it with the application f ? ?. f is the only function that can produce the type S so there is only one choice. We continue in a similar way to pick the left most meta variable until we reach

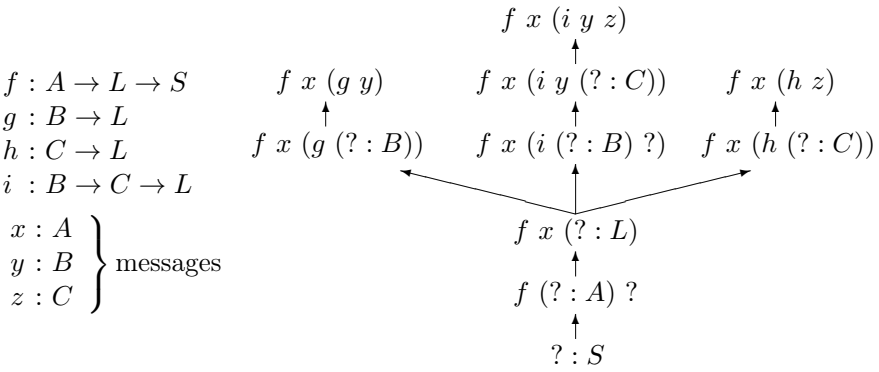


Figure 2: An example abstract syntax and the corresponding choice tree

a leaf in the choice tree which is always a complete term. For convenience on the figure we always annotate the left most meta variable with its type. We see that on the third step there are three possible choices for a function of type L and this creates three branches in the tree. At the end they lead to three different complete expressions. The existing search algorithms will incrementally construct parts of the choice tree by either following a random branch for random generation or by using breadth-first priority traversal for exhaustive search.

Contrary to the existing search strategies, in document planning we do not need all trees but only trees that incorporate a given set of messages. The set of messages we identify with a set of types in the abstract syntax, the actual content of each message will correspond to an expression of the corresponding type. The communicative goal on the other hand, as we said, is nothing else but the initial type from which we start the generation.

In the example the communicative goal would be the type S. If the same grammar is used for natural language generation with distinct goals, then this would mean that different initial types should be used for each goal. As a set of messages we have chosen the types A, B and C. For simplicity, the example includes only one possible value for each type, i.e. x, y and z correspondingly, but this is not the usual situation in real examples. For example in our use case, one possible type of message is `Temperature` with a value which might look like:

```
TemperatureVal 8.0
```

where the actual number will be the temperature as given from the data.

Once we have identified the messages and the initial type, we can now express queries. Each query is a linear logic like term which is a multiplicative conjunction of either:

- a pair of an expression and its type, e.g. $t : A$ where t is the expression and A is a type for messages;

- a negation of a message type, e.g. $\neg A$.

The semantics for a query is to find all expressions which incorporate all non-negated messages but do not include any of the negated messages. Note that the framework does not know a priory which types should be treated as messages and which not. It all depends on the content of the query.

In our toy example, if we execute the query:

$$(x : A) \otimes (y : B)$$

then the types A and B will be treated as messages. From the figure we see that there are two expressions: $f\ x\ (g\ y)$ and $f\ x\ (i\ y\ z)$ that contain them. Note that the term $z : C$ is also included but this is because we did not explicitly forbade the type C. If we want to threat it as a message type then the query should be:

$$(x : A) \otimes (y : B) \otimes \neg C$$

which has only one solution: $f\ x\ (g\ y)$. In general according to the query, we threat types which are included in it as messages and all other types as syntactic categories. For a message type the planner either: always uses the specified value for a positive type or forbids the use of that type completely, when it is used negatively. For types that are not included in the query the planner is free to use arbitrary functions of that type as many times as necessary. Naturally these types act as syntactic categories that are needed to construct a coherent text.

The example also illustrates how aggregation of messages works. If we want to communicate all of the three messages x, y and z then we can execute the query:

$$(x : A) \otimes (y : B) \otimes (z : C)$$

which will retrieve the expression $f\ x\ (i\ y\ z)$. The function i takes as arguments both messages y and z and is thus free to embed them in a single sentence. If on the other hand we have only the message z then we can issue the query:

$$(x : A) \otimes \neg B \otimes (z : C)$$

which will result in $f\ x\ (h\ z)$.

Note that all possible expressions in the example must include the message x. This means that the planning will fail if we do not have that information in the original source. This is an example of a message that must belong to the nucleus of the document and thus no document can be constructed without it. In our use case the city name and the current date are examples of messages of that kind. However, we know that these values always exist because they are part of the input to DarkSky and to the whole system.

The implementation of the document planner on top of the framework's normal search algorithm requires just a modest extension. We still construct the choice tree as in Figure 2 but now along with the current expression, we also carry the current query. Whenever the type of the current meta variable matches one of the positive types in the query then instead of continuing with a free choice from the tree, we just insert the expression from the query. Since every message must be embedded only once in the final expression, we also update the query by replacing the corresponding query term with a negative term for the same type. If, on the other hand, the type of the meta variable matches a negative term then we just abandon the current branch and backtrack to an alternative branch. In this way if we encounter twice a meta variable of the same message type then only the first attempt succeeds. If the type of the current meta variable is not in the query then we can make a free choice constrained only by the structure of the choice tree. The overall search is successful only if in the final query all terms are negative.

The following shows the current state and the corresponding query after every step in the generation of $f\ x\ (i\ y\ z)$:

$? : S$	$(x : A) \otimes (y : B) \otimes (z : C)$
$f\ (? : A)\ ?$	$(x : A) \otimes (y : B) \otimes (z : C)$
$f\ x\ (? : L)$	$\neg A \otimes (y : B) \otimes (z : C)$
$f\ x\ (i\ (? : B)\ ?)$	$\neg A \otimes (y : B) \otimes (z : C)$
$f\ x\ (i\ y\ (? : C))$	$\neg A \otimes \neg B \otimes (z : C)$
$f\ x\ (i\ y\ z)$	$\neg A \otimes \neg B \otimes \neg C$

Note that it is completely possible that for one and the same query, there might be several matching expressions. This is acceptable if the grammar is designed in such a way that all possible expressions convey the same information given the same values for the messages. This might be even desirable if we want some variability in the natural language generation. In that case for instance we can pick a random tree every time when the planner is executed.

4 Implementation

The implementation consists of three parts: a GF grammar, the new document planning algorithm in the GF runtime, and a host Haskell application, which receives weather data, constructs, calls the planner and produces output text. Below are example fragments in English and Russian followed by the corresponding abstract syntax tree.

```
On Sunday, 16 April 2017 at 16:08 in Gothenburg it is snowing.
It is very cold: the temperature is 4.22 °C and it feels
like 0.95 °C.

В воскресение, 16 апреля 2017 16:08 в Гётеборге снежно.
```

Field	Description
temperature	The air temperature in degrees Fahrenheit
cloudCover	The percentage of sky occluded by clouds, between 0 and 1, inclusive.
dewPoint	The dew point in degrees Fahrenheit.
humidity	The relative humidity, between 0 and 1, inclusive.
icon	A machine-readable text summary of this data point, suitable for selecting an icon for display.
ozone	The columnar density of total atmospheric ozone at the given time in Dobson units.
precipIntensity	The intensity (in inches of liquid water per hour) of precipitation occurring at the given time.
precipProbability	The probability of precipitation occurring, between 0 and 1, inclusive.
precipType	The type of precipitation occurring at the given time.
pressure	The sea-level air pressure in millibars.
time	The UNIX time at which this data point begins.
windBearing	The direction that the wind is coming from in degrees, with true north at 0° and progressing clockwise.
windSpeed	The wind speed in miles per hour.

Table 1: Some fields of a JSON-response object as provided by the API of DarkSky. We list those that are used in the current implementation of the text robot. Some of them are optional or belong to a data-point object, and the application handles their faulty absence. All listed fields are transformed to related atomic messages after the data is received, verified, and processed. Descriptions are taken from the API documentation.

```
Очень холодно: температура 4.22 °C и 0.95 °C по ощущениям.

Background (InfoLocation Gothenburg
                         (DayVal "16") April (YearVal "2017")
                         Saturday (TimeVal "16:09")
                         IconClearDay)
           (BSat (InfoTemperature VeryCold
                                  (TemperatureVal 4.22)
                                  (ApparentTemperatureVal 0.95)))
```

The grammar is compiled to a PGF file containing a portable grammar, accompanied with a Haskell interface in an automatically generated source code. Together with the PGF Haskell library [4], the interface facilitates abstract syntax tree manipulations. The library also contains the document planning algorithm — a modified proof search algorithm.

The host application receives, verifies, and processes raw data from DarkSky. If some fields are absent, we mark that by using a negative term in the query. Regarding processing, some received values, for example temperature or wind speed, are translated from imperial units to metric. Some JSON fields that we use in the current implementation are listed in Table 1. Then, values are transformed to atomic messages and supplied to the document planning algorithm, which returns an abstract syntax representation of a weather report ready to be linearised to English or Russian using the standard PGF API.

5 Conclusion

This work has demonstrated that a small-scale modification of a GF normal proof search algorithm could facilitate the document planning phase of a natural language generation. By using randomized search and some redundancy in the abstract syntax of the grammar we could also achieve variability in the generated text. The inherent multilinguality of the framework also allows the same report to be rendered in several languages at once. Also, a feasible structure of an application grammar together with the requirements for its completeness have been outlined and formulated. The weather report case illustrates practical details of the approach and is an evidence of its straightforward implementation.

References

[1] Martin Agfjord, Krasimir Angelov, Per Fredelius, and Svetoslav Marinov. Grammar-based suggestion engine with keyword search. In *Swedish Language Technology Conference*, 2014.

[2] Krasimir Angelov. *The Mechanics of the Grammatical Framework*. PhD thesis, Chalmers University of Technology, 2011.

[3] Krasimir Angelov. *Probability Distributions in Type Theory with Applications in Natural Language Syntax*, pages 279–296. Springer International Publishing, 2017.

[4] Krasimir Angelov, Björn Bringert, and Aarne Ranta. PGF: A Portable Run-Time Format for Type-Theoretical Grammars. *Journal of Logic, Language and Information*, 19:201–228, 2010.

[5] Olga Caprotti and Mika Seppälä. Multilingual delivery of online tests in mathematics. *Proceedings of Online Educa Berlin*, 2006, 2006.

[6] Dana Dannells. *Multilingual text generation from structured formal representations*. PhD in Philosophy, University of Gothenburg. Faculty of Arts, Gothenburg, 2013.

[7] Eva Forsbom. Rhetorical structure theory in natural language generation, Spring 2005.

[8] Thomas Hallgren. The correctness of insertion sort, 2001. Manuscript, Chalmers University.

[9] Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *J. ACM*, 40:143–184, January 1993.

[10] Kristofer Johannisson. *Formal and informal software specifications*. Citeseer, 2005.

[11] William C Mann and Sandra A Thompson. Towards a theory of document structure. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281, 1988.

[12] Per Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, Napoli, 1984.

[13] Per Martin-Löf. Mathematics of infinity. In Per Martin-Löf and Grigori Mints, editors, *Conference on Computer Logic*, volume 417 of *Lecture Notes in Computer Science*, pages 146–197. Springer, 1988.

[14] Bengt Nordström. Towards a theory of document structure. In Yves Bertot, Gerard Huet, Jean-Jacques Levy, and Gordon Plotkin, editors, *From Semantics to Computer Science: Essays in Honor of Gilles Kahn*, pages 265–279. Cambridge University Press, 2008.

[15] Aarne Ranta. The GF resource grammar library. *Linguistic Issues in Language Technology*, 2009.

[16] Aarne Ranta. *Grammatical Framework: Programming with Multilingual Grammars*. CSLI Publications, Stanford, 2011. ISBN-10: 1-57586-626-9 (Paper), 1-57586-627-7 (Cloth).

[17] Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. Cambridge University Press, New York, NY, USA, 2000.

Parsing Logical Grammar: CatLog3*

Glyn Morrill
Department of Computer Science
Universitat Politècnica de Catalunya
Barcelona, Spain.

morrill@cs.upc.edu

Abstract

CatLog3 is a Prolog parser/theorem-prover for (type) logical (categorial) grammar. In such logical grammar, grammar is *reduced* to logic: a string of words is grammatical if and only if an associated logical statement is a theorem. CalLog3 implements a logic extending displacement calculus, a sublinear fragment including as primitive connectives the continuous (Lambek) and discontinuous wrapping connectives of the displacement calculus, additives, 1st order quantifiers, normal modalities, bracket modalities and subexponentials. In this paper we survey how CatLog3 is implemented on the principles of Andreoli's focusing and a generalisation of van Benthem's count-invariance.

1 Introduction

The linguistics that has descended from formal grammar as popularised by Chomsky (1957[6]) has renegaded on formalisation, and discrete computational grammar in the genre of the 1980s has given way to statistical NLP. However, there is a venerable older line of linguistics practicing grammar according to the standards of mathematical logic. The seminal paper in this line is Lambek (1958[19]) which defines a syntactic calculus and proves Cut-elimination for it, but the tradition dates back at least to Bar-Hillel (1953[3]) and Ajdukiewicz (1935[1]). The Lambek calculus is a calculus of concatentation which is free of structural rules. The displacement calculus of Morrill et al. (2011[46]) generalises Lambek calculus with intercalation, containing both continuous and discontinuous connective families, while remaining free of structural rules, and preserving Cut-elimination and its good corollaries: the subformula property, decidability, the finite reading property, and the focusing property. There are several monographs and reference articles on this type logical approach: Moortgat (1988[22]; 1997[24])), Morrill (1994[47];

*This work was partially supported by an ICREA Academia 2012, and MINECO project APCOM (TIN2014-57226-P). Thanks to Oriol Valentín for discussions in relation to this work.

2010[31]; 2011[48]; 2012[34]), Carpenter (1997[5]), Jäger (2005[13]), and Moot and Retoré (2012[27]). The CatLog program series comprises implementations in Prolog of type logical parser/theorem-provers starting from the basis of logic programming of displacement calculus theorem-proving (Morrill 2011[32]):

- CatLog1 (Morrill 2012[33] was based on the method of uniform proof (Miller et al. 1991[20]), and the method of count-invariance for multiplicatives (van Benthem 1991[51]).
- CatLog2 was based on Andreoli's focusing (Andreoli 1992[2]) , and count-invariance for multiplicatives, additives and bracket modalities (Valentín et al. 2013[50]).
- CatLog3 is based on focalisation and count-invariance for multiplicatives, additives, bracket modalities and exponentials (Kuznetov et al. 2017[16]).

In this paper we survey the methods on which the implementation of CatLog3 is based. In Section 2 we describe the primitive connectives of the logical fragment for which parsing/theorem-proving is implemented. In Sections 3 and 4 we discuss focusing and count-invariance respectively. In Section 5 we illustrate in relation to the *Montague Test* (Morrill and Valentín 2016[43]): the task of providing a computational grammar of Montague's (1973[21]) fragment.

2 Displacement logic

The formalism used comprises the connectives of Table 1. The heart of the logic is the displacement calculus of Morrill and Valentín (2010[38]) and Morrill, Valentín and Fadda (2011[46]) made up of twin continuous and discontinuous residuated families of connectives having a pure Gentzen sequent calculus —without labels and free of structural rules— and enjoying Cut-elimination (Valentín 2012[49]). Other primary connectives include additives, 1st order quantifiers, normal (i.e. distributive) modalities, bracket (i.e. nondistributive) modalities, and exponentials.[1] We can draw a clear distinction between the primary connectives, the semantically inactive connectives, and the synthetic connectives; the latter two are abbreviatory and are there for convenience, and to simplify derivation. There are semantically inactive variants of the continuous and discontinuous multiplicatives, and semantically inactive variants of the additives, 1st order quantifiers, and normal modalities.[2] Synthetic connectives (Girard 2011[11]) divide into the continuous

[1] Once Cut-elimination is established, the only challenge to decidability comes from nonlinearity: the contraction rule of the universal exponential, and the infinitary left rule of the existential exponential. In this connection, linguistically the existential left rule is not required; and Morrill and Valentín (2015[41]) introduced displacement logics $\mathbf{Db!_b?}$ and **Db!?** with a relevant modality ! ,with and without bracket conditioning for contaction. Kanovich et al. (2016[15]) prove the undecidability of **Db**!? and in unpublished work announce the undecidability of $\mathbf{Db!_b?}$. But Morrill and Valentín (2015[41]) prove the decidability of a linguistically sufficient 'bracket non-negative' special case of $\mathbf{Db!_b?}$.

[2] For example, the semantically inactive additive conjunction $A \sqcap B\colon \phi$ abbreviates $A \& B\colon (\phi, \phi)$.

	cont. mult.		disc. mult.		add.	qu.	norm. mod.	brack. mod.	exp.	lim. contr. & weak.
primary	/	\	↑	↓	&	⋀	□	$[\,]^{-1}$	!	|
	•		⊙							
	I		J		⊕	⋁	◇	⟨⟩	?	W
sem.	●– –○		○– –●		⊓	∀	■			
inactive variants	◐		◑		◓	◒	⧫			
det.	$\triangleleft^{-1}$		$\triangleright^{-1}$		ˇ					diff.
synth.	◁		▷		ˆ					
nondet.	÷				⇑	⇓				–
synth.	×				◎					

Table 1: Categorial connectives

1.	$\mathcal{F}_i$	::=	$\mathcal{F}_{i+j}/\mathcal{F}_j$	$T(C/B)$	=	$T(B)\to T(C)$	over
2.	$\mathcal{F}_j$	::=	$\mathcal{F}_i\backslash\mathcal{F}_{i+j}$	$T(A\backslash C)$	=	$T(A)\to T(C)$	under
3.	$\mathcal{F}_{i+j}$	::=	$\mathcal{F}_i\bullet\mathcal{F}_j$	$T(A\bullet B)$	=	$T(A)\&T(B)$	continuous product
4.	$\mathcal{F}_0$	::=	I	$T(I)$	=	$\top$	continuous unit
5.	$\mathcal{F}_{i+1}$	::=	$\mathcal{F}_{i+j}\uparrow_k\mathcal{F}_j, 1\le k\le i+j$	$T(C\uparrow_k B)$	=	$T(B)\to T(C)$	circumfix
6.	$\mathcal{F}_j$	::=	$\mathcal{F}_{i+1}\downarrow_k\mathcal{F}_{i+j}, 1\le k\le i+1$	$T(A\downarrow_k C)$	=	$T(A)\to T(C)$	infix
7.	$\mathcal{F}_{i+j}$	::=	$\mathcal{F}_{i+1}\odot_k\mathcal{F}_j, 1\le k\le i+1$	$T(A\odot_k B)$	=	$T(A)\&T(B)$	discontinuous product
8.	$\mathcal{F}_1$	::=	J	$T(J)$	=	$\top$	discontinuous unit
9.	$\mathcal{F}_i$	::=	$\mathcal{F}_i\&\mathcal{F}_i$	$T(A\&B)$	=	$T(A)\&T(B)$	additive conjunction
10.	$\mathcal{F}_i$	::=	$\mathcal{F}_i\oplus\mathcal{F}_i$	$T(A\oplus B)$	=	$T(A)+T(B)$	additive disjunction
11.	$\mathcal{F}_i$	::=	$\bigwedge V\mathcal{F}_i$	$T(\bigwedge vA)$	=	$F\to T(A)$	1st order univ. qu.
12.	$\mathcal{F}_i$	::=	$\bigvee V\mathcal{F}_i$	$T(\bigvee vA)$	=	$F\&T(A)$	1st order exist. qu.
13.	$\mathcal{F}_i$	::=	$\Box\mathcal{F}_i$	$T(\Box A)$	=	$\mathbf{L}T(A)$	universal modality
14.	$\mathcal{F}_i$	::=	$\Diamond\mathcal{F}_i$	$T(\Diamond A)$	=	$\mathbf{M}T(A)$	existential modality
15.	$\mathcal{F}_i$	::=	$[\,]^{-1}\mathcal{F}_i$	$T([\,]^{-1}A)$	=	$T(A)$	univ. bracket modality
16.	$\mathcal{F}_i$	::=	$\langle\rangle\mathcal{F}_i$	$T(\langle\rangle A)$	=	$T(A)$	exist. bracket modality
17.	$\mathcal{F}_0$	::=	$!\mathcal{F}_0$	$T(!A)$	=	$T(A)$	universal exponential
18.	$\mathcal{F}_0^\circ$	::=	$?\mathcal{F}_0^\circ$	$T(?A)$	=	$T(A)^+$	existential exponential

Table 2: Syntactic types of **DA1S4b!$_\mathbf{b}$?**

and discontinuous deterministic (unary) synthetic connectives, and the continuous and discontinuous nondeterministic (binary) synthetic connectives.[3]

2.1 Syntactic types

The syntactic types of displacement logic are sorted $\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_2, \ldots$ according to the number of points of discontinuity $0, 1, 2, \ldots$ their expressions contain. Each type predicate letter has a sort and an arity which are naturals, and a corresponding semantic type. Assuming ordinary terms to be already given, where P is a type predicate letter of sort i and arity n and $t_1, \ldots, t_n$ are terms, $Pt_1 \ldots t_n$ is an (atomic) type of sort i of the corresponding semantic type. Compound types of **DA1S4b!$_\mathbf{b}$?** are formed as illustrated in Table 2, and the structure preserving semantic type map T associates these with semantic types.

2.2 Gentzen sequent calculus

We use a Gentzen sequent presentation standard from Gentzen (1934[9]) and Lambek (1958[19]). In Gentzen sequent antecedents for displacement logic with bracket modalities (structural inhibition) and exponentials (structural facilitation) there are also bracket constructors and 'stoups'.

Stoups (cf. the linear logic of Girard 2011[11]) (ζ) are stores read as multisets for re-usable (nonlinear) resources which appear at the left of a configuration marked off by a semicolon (when the stoup is empty the semicolon may be ommited). The stoup of linear logic is for resources which can be contracted (copied)

[3] For example, the nondeterministic continuous division $B\div A$ abbreviates $(A\backslash B)\sqcap(B/A)$.

or weakened (deleted). By contrast, our stoup is for a linguistically motivated variant of contraction, and does not allow weakening. Furthermore, whereas linear logic is commutative, our logic is in general noncommutative and here the stoup is used for resources which are also commutative. A configuration together with a stoup is a *zone* (Ξ). The bracket constructor applies not to a configuration alone but to a configuration with a stoup, i.e a zone: reusable resources are specific to their domain. Stoups $\mathcal{S}$ and configurations $\mathcal{O}$ are defined by ($\emptyset$ is the empty stoup; Λ is the empty configuration; the *separator* 1 marks points of discontinuity):[4]

$$\begin{array}{lrl} (1) & \mathcal{S} & ::= \quad \emptyset \mid \mathcal{F}_0, \mathcal{S} \\ & \mathcal{O} & ::= \quad \Lambda \mid \mathcal{T}, \mathcal{O} \\ & \mathcal{T} & ::= \quad 1 \mid \mathcal{F}_0 \mid \mathcal{F}_{i>0}\{\underbrace{\mathcal{O} : \ldots : \mathcal{O}}_{i\,\mathcal{O}'\mathrm{s}}\} \mid [\mathcal{S}; \mathcal{O}] \end{array}$$

For a type A, its sort $s(A)$ is the i such that $A \in \mathcal{F}_i$. For a configuration Γ, its sort $s(\Gamma)$ is $|\Gamma|_1$, i.e. the number of points of discontinuity 1 which it contains. Sequents are of the form:

(2) $\mathcal{S}; \mathcal{O} \Rightarrow \mathcal{F}$ such that $s(\mathcal{O}) = s(\mathcal{F})$

The figure $\overrightarrow{A}$ of a type A is defined by:

$$(3)\quad \overrightarrow{A} = \begin{cases} A & \text{if } s(A) = 0 \\ A\{\underbrace{1 : \ldots : 1}_{s(A)\ 1'\mathrm{s}}\} & \text{if } s(A) > 0 \end{cases}$$

Where Γ is a configuration of sort i and $\Delta_1, \ldots, \Delta_i$ are configurations, the *fold* $\Gamma \otimes \langle \Delta_1 : \ldots : \Delta_i \rangle$ is the result of replacing the successive 1's in Γ by $\Delta_1, \ldots, \Delta_i$ respectively. Where Γ is of sort i, the hyperoccurrence notation $\Delta\langle\Gamma\rangle$ abbreviates $\Delta_0(\Gamma \otimes \langle \Delta_1 : \ldots : \Delta_i \rangle)$, i.e. a context configuration Δ (which is externally Δ_0 and internally $\Delta_1, \ldots, \Delta_i$) with a potentially discontinuous distinguished subconfiguration Γ (continuous if $i = 0$, discontinuous if $i > 0$). Where Δ is a configuration of sort $i > 0$ and Γ is a configuration, the kth *metalinguistic intercalation* $\Delta \mid_k \Gamma$, $1 \leq k \leq i$, is given by:

$$(4)\quad \Delta \mid_k \Gamma =_{df} \Delta \otimes \langle \underbrace{1 : \ldots : 1}_{k-1\ 1'\mathrm{s}} : \Gamma : \underbrace{1 : \ldots : 1}_{i-k\ 1'\mathrm{s}} \rangle$$

i.e. $\Delta \mid_k \Gamma$ is the configuration resulting from replacing by Γ the kth separator in Δ.

2.3 Rules and linguistic applications

A semantically labelled sequent is a sequent in which the antecedent type occurrences $A_1, \ldots, A_n$ are labelled by distinct variables $x_1, \ldots, x_n$ which are of types

[4] Note that only types of sort 0 can go into the stoup; reusable types of other sorts would not preserve the sequent antecedent-succedent sort equality under contraction or expansion: $0 + 0 = 0$, but $i + i \neq i$ for $i > 0$.

$T(A_1), \ldots, T(A_n)$ respectively, and the succedent type A is labelled by a term of type $T(A)$ with free variables drawn from $x_1, \ldots, x_n$. In this section we give the semantically labelled Gentzen sequent rules for the connectives of **DA1S4b!$_\flat$?**, and indicate some linguistic applications.

1. $$\frac{\zeta_1; \Gamma \Rightarrow B\colon \psi \qquad \Xi(\zeta_2; \Delta\langle \overrightarrow{C}\colon z\rangle) \Rightarrow D\colon \omega}{\Xi(\zeta_1 \uplus \zeta_2; \Delta\langle \overrightarrow{C/B}\colon x, \Gamma\rangle) \Rightarrow D\colon \omega\{(x\ \psi)/z\}}\ /L \qquad \frac{\zeta; \Gamma, \overrightarrow{B}\colon y \Rightarrow C\colon \chi}{\zeta; \Gamma \Rightarrow C/B\colon \lambda y\chi}\ /R$$

2. $$\frac{\zeta_1; \Gamma \Rightarrow A\colon \phi \qquad \Xi(\zeta_2; \Delta\langle \overrightarrow{C}\colon z\rangle) \Rightarrow D\colon \omega}{\Xi(\zeta_1 \uplus \zeta_2; \Delta\langle \Gamma, \overrightarrow{A\backslash C}\colon y\rangle) \Rightarrow D\colon \omega\{(y\ \phi)/z\}}\ \backslash L \qquad \frac{\zeta; \overrightarrow{A}\colon x, \Gamma \Rightarrow C\colon \chi}{\zeta; \Gamma \Rightarrow A\backslash C\colon \lambda x\chi}\ \backslash R$$

3. $$\frac{\Xi\langle \overrightarrow{A}\colon x, \overrightarrow{B}\colon y\rangle \Rightarrow D\colon \omega}{\Xi\langle \overrightarrow{A \bullet B}\colon z\rangle \Rightarrow D\colon \omega\{\pi_1 z/x, \pi_2 z/y\}}\ \bullet L \qquad \frac{\zeta_1; \Delta \Rightarrow A\colon \phi \qquad \zeta_2; \Gamma \Rightarrow B\colon \psi}{\zeta_1 \uplus \zeta_2; \Delta, \Gamma \Rightarrow A \bullet B\colon (\phi, \psi)}\ \bullet R$$

4. $$\frac{\Xi\langle \Lambda\rangle \Rightarrow A\colon \phi}{\Xi\langle \overrightarrow{I}\colon x\rangle \Rightarrow A\colon \phi}\ IL \qquad \frac{}{\emptyset; \Lambda \Rightarrow I\colon 0}\ IR$$

Figure 1: Lambek multiplicatives

The continuous multiplicatives, the Lambek connectives of Lambek (1958[19]; 1988[18]), Figure 1, defined in relation to concatenation/appending, are the basic means of categorial categorization and subcategorization. Note that here and throughout the active types in antecedents are figures (vectorial) whereas those in succedents are not; intuitively this is because antecedents are structured but succedents are not. The directional divisions over, /, and under, \, are exemplified by assignments such as *the*: N/CN for *the man*: N and *sings*: $N\backslash S$ for *John sings*: S, and *loves*: $(N\backslash S)/N$ for *John loves Mary*: S.

The discontinuous multiplicatives of Figure 2, the displacement connectives, Morrill and Valentín (2010[38]), Morrill et al. (2011[46]), are defined in relation to intercalation/plugging. When the value of the k subindex indicates the first (leftmost) point of discontinuity it may be omitted, i.e. it defaults to 1. Circumfixation, $\uparrow$, is exemplified by a discontinuous particle verb assignment such as *calls*+1+*up*: $(N\backslash S)\uparrow N$ for *Mary calls John up*: S, and infixation, $\downarrow$, and circumfixation together are exemplified by a quantifier phrase assignment of the form *everyone*: $(S\uparrow N)\downarrow S$ simulating Montague's S14 treatment of quantifying in; see Section 5.

In relation to the multiplicative rules, notice how the stoup is distributed reading bottom-up from conclusions to premise: it is partitioned between the two premises in the case of binary rules, copied to the premise in the case of unary rules, and empty in the case of nullary rules (axioms).

The additives of Figure 3, Lambek (1961[17]), Morrill (1990[28]), Kanazawa (1992[14]), have application to polymorphism. For example the additive conjunc-

5. $$\frac{\zeta_1;\Gamma \Rightarrow B:\psi \qquad \Xi(\zeta_2;\Delta\langle\overrightarrow{C}:z\rangle) \Rightarrow D:\omega}{\Xi(\zeta_1 \uplus \zeta_2;\Delta\langle\overrightarrow{C\uparrow_k B}:x\,|_k\,\Gamma\rangle) \Rightarrow D:\omega\{(x\,\psi)/z\}}\uparrow_k L \qquad \frac{\zeta;\Gamma\,|_k\,\overrightarrow{B}:y \Rightarrow C:\chi}{\zeta;\Gamma \Rightarrow C\uparrow_k B:\lambda y\chi}\uparrow_k R$$

6. $$\frac{\zeta_1;\Gamma \Rightarrow A:\phi \qquad \Xi(\zeta_2;\Delta\langle\overrightarrow{C}:z\rangle) \Rightarrow D:\omega}{\Xi(\zeta_1 \uplus \zeta_2;\Delta\langle\Gamma\,|_k\,\overrightarrow{A\downarrow_k C}:y\rangle) \Rightarrow D:\omega\{(y\,\phi)/z\}}\downarrow_k L \qquad \frac{\zeta;\overrightarrow{A}:x\,|_k\,\Gamma \Rightarrow C:\chi}{\zeta;\Gamma \Rightarrow A\downarrow_k C:\lambda x\chi}\downarrow_k R$$

7. $$\frac{\Xi\langle\overrightarrow{A}:x\,|_k\,\overrightarrow{B}:y\rangle \Rightarrow D:\omega}{\Xi\langle\overrightarrow{A\odot_k B}:z\rangle \Rightarrow D:\omega\{\pi_1 z/x,\pi_2 z/y\}}\odot_k L \qquad \frac{\zeta_1;\Delta \Rightarrow A:\phi \qquad \zeta_2;\Gamma \Rightarrow B:\psi}{\zeta_1 \uplus \zeta_2;\Delta\,|_k\,\Gamma \Rightarrow A\odot_k B:(\phi,\psi)}\odot_k R$$

8. $$\frac{\Xi\langle 1\rangle \Rightarrow A:\phi}{\Xi\langle\overrightarrow{J}:x\rangle \Rightarrow A:\phi}JL \qquad \frac{}{\emptyset;1 \Rightarrow J:0}JR$$

Figure 2: Displacement multiplicatives

9. $$\frac{\Xi\langle\overrightarrow{A}:x\rangle \Rightarrow C:\chi}{\Xi\langle\overrightarrow{A\&B}:z\rangle \Rightarrow C:\chi\{\pi_1 z/x\}}\&L_1 \qquad \frac{\Xi\langle\overrightarrow{B}:y\rangle \Rightarrow C:\chi}{\Xi\langle\overrightarrow{A\&B}:z\rangle \Rightarrow C:\chi\{\pi_2 z/y\}}\&L_2$$

$$\frac{\Xi \Rightarrow A:\phi \qquad \Xi \Rightarrow B:\psi}{\Xi \Rightarrow A\&B:(\phi,\psi)}\&R$$

10. $$\frac{\Xi\langle\overrightarrow{A}:x\rangle \Rightarrow C:\chi_1 \qquad \Xi\langle\overrightarrow{B}:y\rangle \Rightarrow C:\chi_2}{\Xi\langle\overrightarrow{A\oplus B}:z\rangle \Rightarrow C:z \to x.\chi_1;y.\chi_2}\oplus L$$

$$\frac{\Xi \Rightarrow A:\phi}{\Xi \Rightarrow A\oplus B:\iota_1\phi}\oplus R_1 \qquad \frac{\Xi \Rightarrow B:\psi}{\Xi \Rightarrow A\oplus B:\iota_2\psi}\oplus R_2$$

Figure 3: Additives

tion & can be used for *rice*: $N\&CN$ as in *rice grows*: S and *the rice grows*: S,[5] and the additive disjunction $\oplus$ can be used for *is*: $(N\backslash S)/(N\oplus(CN/CN))$ as in *Tully is Cicero*: S and *Tully is humanist*: S. The additive disjunction can be used together with the continuous unit to express the optionality of a complement as in *eats*: $(N\backslash S)/(N\oplus I)$ for *John eats fish*: S and *John eats*: S.[6]

Notice how the stoup is identical in conclusions and premises of additive rules.

[5]Note the computational advantage of this approach over assuming an empty determiner: if empty operators were allowed they could occur any number of times in any positions.

[6]Note the advantage of this over simply listing intransitive and transitive lexical entries: empirically the latter does not capture the generalisation that in both cases the verb *eats* combines with a subject to the left, and computationally every lexical ambiguity doubles the lexical insertion search space. Appeal to lexical ambiguity constitutes resignation from the capture of generalisations and is at best a promissory solution, unless there is true ambiguity.

$$11.\quad \frac{\Xi\langle\overrightarrow{A[t/v]}\colon x\rangle \Rightarrow B\colon\psi}{\Xi\langle\overrightarrow{\bigwedge vA}\colon z\rangle \Rightarrow B\colon\psi\{(z\ t)/x\}}\bigwedge L \qquad \frac{\Xi \Rightarrow A[a/v]\colon\phi}{\Xi \Rightarrow \bigwedge vA\colon\lambda v\phi}\bigwedge R^{\dagger}$$

$$12.\quad \frac{\Xi\langle\overrightarrow{A[a/v]}\colon x\rangle \Rightarrow B\colon\psi}{\Xi\langle\overrightarrow{\bigvee vA}\colon z\rangle \Rightarrow B\colon\psi\{\pi_2 z/x\}}\bigvee L^{\dagger} \qquad \frac{\Xi \Rightarrow A[t/v]\colon\phi}{\Xi \Rightarrow \bigvee vA\colon(t,\phi)}\bigvee R$$

Figure 4: Quantifiers, where † indicates that there is no a in the conclusion

The quantifiers of Figure 4, Morrill (1994[47]), have application to features. For example, singular and plural number in *sheep*: $\bigwedge nCNn$ for *the sheep grazes*: S and *the sheep graze*: S. And for a past, present or future tense finite sentence complement we can have *said*: $(N\backslash S)/\bigvee tS\,f(t)$ in *John said Mary walked*: S, *John said Mary walks*: S and *John said Mary will walk*: S.

Notice how the stoup is identical in conclusion and premise in each quantifier rule.

$$13.\quad \frac{\Xi\langle\overrightarrow{A}\colon x\rangle \Rightarrow B\colon\psi}{\Xi\langle\overrightarrow{\Box A}\colon z\rangle \Rightarrow B\colon\psi\{^{\vee}z/x\}}\Box L \qquad \frac{\boxtimes\Xi \Rightarrow A\colon\phi}{\boxtimes\Xi \Rightarrow \Box A\colon{}^{\wedge}\phi}\Box R$$

$$14.\quad \frac{\boxtimes\Xi\langle\overrightarrow{A}\colon x\rangle \Rightarrow \diamondsuit B\colon\psi}{\boxtimes\Xi\langle\overrightarrow{\diamondsuit A}\colon z\rangle \Rightarrow \diamondsuit B\colon\psi\{^{\cup}z/x\}}\diamondsuit L \qquad \frac{\Xi \Rightarrow A\colon\phi}{\Xi \Rightarrow \diamondsuit A\colon{}^{\cap}\phi}\diamondsuit R$$

Figure 5: Normal modalities, where $\boxtimes/\diamondsuit$ marks a structure all the types of which have main connective a box/diamond

With respect to the (**S4**) normal modalities of Figure 5, the universal (Morrill 1990[29]) has application to intensionality. For example, for a propositional attitude verb such as *believes* we can assign type $\Box((N\backslash S)/\Box S)$ with a modality outermost since the word has a sense, and a modality on the first argument but not the second, since the sentential complement is an intensional domain, but not the subject. The modalities are in the categorial type, distinctly from, but in relation to, the logical interpretation of the propositional attitude verb. The $\Box$ Right rule is semantically interpreted by intensionalisation $^{\wedge}$ and the $\Box$ Left rule is semantically interpreted by extensionalisation $^{\vee}$ in such a way that the Curry-Howard correspondence for the modality yields the law of down-up cancellation (Dowty et al. 1981[7]): $^{\vee\wedge}\phi = \phi$.

Notice how the stoup is identical in conclusion and premise in each normal modality rule.

The bracket modalities of Figure 6, Morrill (1992[30]) and Moortgat 1995[23]), have application to nonassociativity and syntactical domains such as extraction

15. $$\frac{\Xi\langle\overrightarrow{A}\colon x\rangle \Rightarrow B\colon\psi}{\Xi\langle\overrightarrow{[\,]^{-1}A}\colon x]\rangle \Rightarrow B\colon\psi}\,[\,]^{-1}L \qquad \frac{[\Xi] \Rightarrow A\colon\phi}{\Xi \Rightarrow [\,]^{-1}A\colon\phi}\,[\,]^{-1}R$$

16. $$\frac{\Xi\langle[\overrightarrow{A}\colon x]\rangle \Rightarrow B\colon\psi}{\Xi\langle\overrightarrow{\langle\rangle A}\colon x\rangle \Rightarrow B\colon\psi}\,\langle\rangle L \qquad \frac{\Xi \Rightarrow A\colon\phi}{[\Xi] \Rightarrow \langle\rangle A\colon\phi}\,\langle\rangle R$$

Figure 6: Bracket modalities

islands and prosodic phrases. For example, single bracketing for weak islands: *walks*: $\langle\rangle N\backslash S$ for the subject condition, and *without*: $[\,]^{-1}(VP\backslash VP)/VP$ for the adverbial island constraint; and double bracketing for strong islands of the kind *and*: $(S\backslash[\,]^{-1}[\,]^{-1}S)/S$ for the coordinate structure constraint.

Notice how the stoup is identical in conclusions and premises of bracket modality rules.

17. $$\frac{\Xi(\zeta \uplus \{A\colon x\}; \Gamma_1, \Gamma_2) \Rightarrow B\colon\psi}{\Xi(\zeta; \Gamma_1, !A\colon x, \Gamma_2) \Rightarrow B\colon\psi}\,!L \qquad \frac{\zeta; \Lambda \Rightarrow A\colon\phi}{\zeta; \Lambda \Rightarrow\, !A\colon\phi}\,!R$$

$$\frac{\Xi(\zeta; \Gamma_1, A\colon x, \Gamma_2) \Rightarrow B\colon\psi}{\Xi(\zeta \uplus \{A\colon x\}; \Gamma_1, \Gamma_2) \Rightarrow B\colon\psi}\,!P$$

$$\frac{\Xi(\zeta_1 \uplus \zeta_2 \uplus \{A\colon x\}; \Gamma_1, [\zeta_2 \uplus \{A\colon y\}; \Gamma_2], \Gamma_3) \Rightarrow B\colon\psi}{\Xi(\zeta_1 \uplus \zeta_2 \uplus \{A\colon x\}; \Gamma_1, \Gamma_2, \Gamma_3) \Rightarrow B\colon\psi\{x/y\}}\,!C$$

18. $$\frac{\Xi(A\colon x_1) \Rightarrow B\colon\psi([x_1]) \qquad \Xi(A\colon x_1, A\colon x_2) \Rightarrow B\colon\psi([x_1, x_2]) \qquad \ldots}{\Xi(?A\colon x) \Rightarrow B\colon\psi(x)}\,?L$$

$$\frac{\Xi \Rightarrow A\colon\phi}{\Xi \Rightarrow ?A\colon[\phi]}\,?R \qquad \frac{\zeta; \Gamma \Rightarrow A\colon\phi \qquad \zeta'; \Delta \Rightarrow ?A\colon\psi}{\zeta \uplus \zeta'; \Gamma, \Delta \Rightarrow ?A\colon[\phi|\psi]}\,?M$$

Figure 7: Exponentials

Finally, there is nonlinearity. The universal exponential of Figure 7, Girard (1987[10]), Barry, Hepple, Leslie and Morrill (1991[4]), Morrill (1994[47]), Morrill and Valentín (2015[41]), and Morrill (2017[35]), has application to extraction including parasitic extraction. In the formulation here $!L$ moves the operand of a universal exponential (e.g. the hypothetical subtype of relativisation) into the stoup, where it will percolate as commented for the above rules. From there it can be copied into the stoup of a newly-created bracketed domain by the contraction rule $!C$ (producing a parasitic gap), and it can be moved into any position in the matrix configuration of its zone by $!P$ (producing a nonparasitic or host gap).

Using the universal exponential, !, for which contraction induces island brackets, we can assign a relative pronoun type *that*: $(CN\backslash CN)/(S/!N)$ allowing parasitic extraction such as *paper that John filed without reading*: *CN*, where parasitic gaps can appear only in (weak) islands, but can be iterated in subislands, for example, *man who the fact that the friends of admire without praising surprises*. Crucially, in the linguistic formulation ! does not have weakening, i.e. deletion, since, e.g., the body of a relative clause *must* contain a gap: **man who John loves Mary*.

The existential exponential ? has application to iterated coordination (Morrill 1994[47]; Morrill and Valentín 2015[41]) and (unboundedly iterated) *respectively* (Morrill and Valentín 2016[44]). Using the existential exponential, ?, we can assign a coordinator type *and*: $(?N\backslash N)/N$ allowing iterated coordination as in *John, Bill, Mary and Suzy*: *N*, or *and*: $(?(S/N)\backslash(S/N))/(S/N)$ for *John likes Mary dislikes, and Bill hates, London* (iterated right node raising), and so on.

In relation to the rest of the primary connectives: the limited contraction | of Jäger (2005[13]) has application to anaphora and the limited weakening *W* of Morrill and Valentín (2014[40]) has application to words as types. The remaining, semantically inactive, connectives listed here were introduced as follows. Semantically inactive multiplicatives {•–, –∘, ∘–, –•, ◐, ◑, ⍏, ⍖, ⍐, ⍗, ◓, ◒}: Morrill and Valentín (2014[40]). Semantically inactive additives {⊓, ⊔}: Morrill (1994[47]). Semantically inactive first-order quantifiers {∀, ∃}: Morrill (1994[47]). Semantically inactive normal modalities {■, ♦}: Hepple (1990[12]), Morrill (1994[47]). The rules for semantically inactive variants are the same as those for the semantically active versions syntactically, but have the same label on premises and conclusions semantically.[7]

3 Focusing

Spurious ambiguity is the phenomenon whereby distinct derivations in grammar may assign the same structural reading, resulting in redundancy in the parse search space and inefficiency in parsing. Understanding the problem depends on identifying the essential mathematical structure of derivations. This is trivial in the case of context free grammar, where the parse structures are ordered trees; in the case of type logical categorial grammar, the parse structures are proof nets. However, with respect to multiplicatives intrinsic proof nets have not yet been given for displacement calculus (but see Morrill and Fadda (2008[36], Fadda 2010[8], and Moot 2014[25], 2016[26]) In this context CatLog3 approaches spurious ambiguity by means of Andreoli's (1982[2]) proof-theoretic technique of focalisation, which engenders a substantial reduction of spurious ambiguity.

[7]The synthetic connectives are: left and right projection and injection $\{\triangleleft^{-1}, \triangleright^{-1}, \triangleleft, \triangleright\}$, Morrill, Fadda and Valentín (2009[45]); split and bridge {ˇ, ˆ}, Morrill and Merenciano (1996[37]); continuous and discontinuous nondeterministic multiplicatives {÷, ×, ⇑, ⇓, ⊚}, Morrill, Valentín and Fadda (2011[46]). The difference operator – of Morrill and Valentín (2014[39]) has application to linguistic exceptions.

$$\frac{\overrightarrow{A}\colon x, \Gamma \Rightarrow C\colon \chi \quad \neg\ \mathbf{foc}}{\Gamma \Rightarrow A\backslash C\colon \lambda x\chi \quad \neg\ \mathbf{foc} \wedge\ \mathbf{rev}}\ \backslash R \qquad \frac{\Gamma, \overrightarrow{B}\colon y \Rightarrow C\colon \chi \quad \neg\ \mathbf{foc}}{\Gamma \Rightarrow C/B\colon \lambda y\chi \quad \neg\ \mathbf{foc} \wedge\ \mathbf{rev}}\ /R$$

$$\frac{\Delta\langle \overrightarrow{A}\colon x, \overrightarrow{B}\colon y\rangle \Rightarrow D\colon \omega \quad \neg\ \mathbf{foc}}{\Delta\langle \overrightarrow{A \bullet B}\colon z\rangle \Rightarrow D\colon \omega\{\pi_1 z/x, \pi_2 z/y\} \quad \neg\ \mathbf{foc} \wedge\ \mathbf{rev}}\ \bullet L$$

$$\frac{\Delta\langle\Lambda\rangle \Rightarrow A\colon \phi \quad \neg\ \mathbf{foc}}{\Delta\langle \overrightarrow{I}\colon x\rangle \Rightarrow A\colon \phi \quad \neg\ \mathbf{foc} \wedge\ \mathbf{rev}}\ IL$$

$$\frac{\overrightarrow{A}\colon x\,|_k\,\Gamma \Rightarrow C\colon \chi \quad \neg\ \mathbf{foc}}{\Gamma \Rightarrow A{\downarrow_k}C\colon \lambda x\chi \quad \neg\ \mathbf{foc} \wedge\ \mathbf{rev}}\ {\downarrow_k}R \qquad \frac{\Gamma\,|_k\,\overrightarrow{B}\colon y \Rightarrow C\colon \chi \quad \neg\ \mathbf{foc}}{\Gamma \Rightarrow C{\uparrow_k}B\colon \lambda y\chi \quad \neg\ \mathbf{foc} \wedge\ \mathbf{rev}}\ {\uparrow_k}R$$

$$\frac{\Delta\langle \overrightarrow{A}\colon x\,|_k\,\overrightarrow{B}\colon y\rangle \Rightarrow D\colon \omega \quad \neg\ \mathbf{foc}}{\Delta\langle \overrightarrow{A \odot_k B}\colon z\rangle \Rightarrow D\colon \omega\{\pi_1 z/x, \pi_2 z/y\} \quad \neg\ \mathbf{foc} \wedge\ \mathbf{rev}}\ \odot_k L$$

$$\frac{\Delta\langle 1\rangle \Rightarrow A\colon \phi \quad \neg\ \mathbf{foc}}{\Delta\langle \overrightarrow{J}\colon x\rangle \Rightarrow A\colon \phi \quad \neg\ \mathbf{foc} \wedge\ \mathbf{rev}}\ JL$$

Figure 8: Reversible multiplicative rules

In focalisation, *situated* (in the antecedent of a sequent, input, $^\bullet/$ in the succedent of a sequent, output, $^\circ$) non-atomic types are classified as of *reversible/negative* or *irreversible/positive polarity* according as their associated rule is reversible or not. There are alternating phases of don't-care nondeterministic negative rule application, and positive rule application locking on to *focalised* formulas. Given a sequent with no occurrences of negative formulas, one chooses a positive formula as principal formula (which is boxed; we say it is focalised) and applies proof search to its subformulas while these remain positive. When one finds a negative formula or a literal, invertible rules are applied in a don't care nondeterminitic fashion until no longer possible, when another positive formula is chosen, and so on. CatLog3 can be set to focus all atoms in the input (as in the example at the end) or in the output, i.e. it implements uniform *bias*.

A sequent is either unfocused and as before, or else focused and has exactly one type boxed. This is the focused type. The focalised logical rules for displacement calculus are given in Figures 8–12. Sequents are accompanied by *judgements*: focalised or not focalised and reversible or not reversible.[8] The completeness of this focalisation, together with additives, is proved in Morrill and Valentín (2015[42]). The completeness of focalisation for other connectives of CatLog3 is a topic of ongoing research.

[8]This idea is due to Oriol Valentín.

$$\frac{\Gamma \Rightarrow \boxed{P}\!:\phi \quad \mathbf{foc} \wedge \neg\, \mathbf{rev} \qquad \Delta\langle \overrightarrow{\boxed{Q}}\!: z\rangle \Rightarrow D\!:\omega \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}}{\Delta\langle \Gamma, \overrightarrow{\boxed{P \backslash Q}}\!: y\rangle \Rightarrow D\!:\omega\{(y\ \phi)/z\} \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}}\ \backslash L$$

$$\frac{\Gamma \Rightarrow \boxed{P_1}\!:\phi \quad \mathbf{foc} \wedge \neg\, \mathbf{rev} \qquad \Delta\langle \overrightarrow{P_2}\!: z\rangle \Rightarrow D\!:\omega \quad \neg\, \mathbf{foc} \wedge ?P_2\ \mathbf{rev}}{\Delta\langle \Gamma, \overrightarrow{\boxed{P_1 \backslash P_2}}\!: y\rangle \Rightarrow D\!:\omega\{(y\ \phi)/z\} \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}}\ \backslash L$$

$$\frac{\Gamma \Rightarrow Q_1\!:\phi \quad \neg\, \mathbf{foc} \wedge ?Q_1\ \mathbf{rev} \qquad \Delta\langle \overrightarrow{\boxed{Q_2}}\!: z\rangle \Rightarrow D\!:\omega \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}}{\Delta\langle \Gamma, \overrightarrow{\boxed{Q_1 \backslash Q_2}}\!: y\rangle \Rightarrow D\!:\omega\{(y\ \phi)/z\} \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}}\ \backslash L$$

$$\frac{\Gamma \Rightarrow Q\!:\phi \quad \neg\, \mathbf{foc} \wedge ?Q\ \mathbf{rev} \qquad \Delta\langle \overrightarrow{P}\!: z\rangle \Rightarrow D\!:\omega \quad \neg\, \mathbf{foc} \wedge ?P\ \mathbf{rev}}{\Delta\langle \Gamma, \overrightarrow{\boxed{Q \backslash P}}\!: y\rangle \Rightarrow D\!:\omega\{(y\ \phi)/z\} \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}}\ \backslash L$$

$$\frac{\Gamma \Rightarrow \boxed{P}\!:\psi \quad \mathbf{foc} \wedge \neg\, \mathbf{rev} \qquad \Delta\langle \overrightarrow{\boxed{Q}}\!: z\rangle \Rightarrow D\!:\omega \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}}{\Delta\langle \overrightarrow{\boxed{Q / P}}\!: x, \Gamma\rangle \Rightarrow D\!:\omega\{(x\ \psi)/z\} \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}}\ /L$$

$$\frac{\Gamma \Rightarrow Q_1\!:\psi \quad \neg\, \mathbf{foc} \wedge ?Q_1\ \mathbf{rev} \qquad \Delta\langle \overrightarrow{\boxed{Q_2}}\!: z\rangle \Rightarrow D\!:\omega \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}}{\Delta\langle \overrightarrow{\boxed{Q_2 / Q_1}}\!: x, \Gamma\rangle \Rightarrow D\!:\omega\{(x\ \psi)/z\} \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}}\ /L$$

$$\frac{\Gamma \Rightarrow \boxed{P_1}\!:\psi \quad \mathbf{foc} \wedge \neg\, \mathbf{rev} \qquad \Delta\langle \overrightarrow{P_2}\!: z\rangle \Rightarrow D\!:\omega \quad \neg\, \mathbf{foc} \wedge ?P_2\ \mathbf{rev}}{\Delta\langle \overrightarrow{\boxed{P_2 / P_1}}\!: x, \Gamma\rangle \Rightarrow D\!:\omega\{(x\ \psi)/z\} \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}}\ /L$$

$$\frac{\Gamma \Rightarrow Q\!:\psi \quad \neg\, \mathbf{foc} \wedge ?Q\ \mathbf{rev} \qquad \Delta\langle \overrightarrow{P}\!: z\rangle \Rightarrow D\!:\omega \quad \neg\, \mathbf{foc} \wedge ?P\ \mathbf{rev}}{\Delta\langle \overrightarrow{\boxed{P / Q}}\!: x, \Gamma\rangle \Rightarrow D\!:\omega\{(x\ \psi)/z\} \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}}\ /L$$

Figure 9: Left irreversible continuous multiplicative rules

$$\frac{\Gamma \Rightarrow \boxed{P} : \phi \quad \mathbf{foc} \wedge \neg\, \mathbf{rev} \qquad \Delta\langle \overrightarrow{\boxed{Q}} : z\rangle \Rightarrow D : \omega \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}}{\Delta\langle \Gamma \mid_k \overrightarrow{\boxed{P \downarrow_k Q}} : y\rangle \Rightarrow D : \omega\{(y\ \phi)/z\} \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}} \downarrow_k L$$

$$\frac{\Gamma \Rightarrow \boxed{P_1} : \phi \quad \mathbf{foc} \wedge \neg\, \mathbf{rev} \qquad \Delta\langle \overrightarrow{P_2} : z\rangle \Rightarrow D : \omega \quad \neg\, \mathbf{foc} \wedge ?P_2\ \mathbf{rev}}{\Delta\langle \Gamma \mid_k \overrightarrow{\boxed{P_1 \downarrow_k P_2}} : y\rangle \Rightarrow D : \omega\{(y\ \phi)/z\} \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}} \downarrow_k L$$

$$\frac{\Gamma \Rightarrow Q_1 : \phi \quad \neg\, \mathbf{foc} \wedge ?Q_1\ \mathbf{rev} \qquad \Delta\langle \overrightarrow{\boxed{Q_2}} : z\rangle \Rightarrow D : \omega \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}}{\Delta\langle \Gamma \mid_k \overrightarrow{\boxed{Q_1 \downarrow_k Q_2}} : y\rangle \Rightarrow D : \omega\{(y\ \phi)/z\} \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}} \downarrow_k L$$

$$\frac{\Gamma \Rightarrow Q : \phi \quad \neg\, \mathbf{foc} \wedge ?Q\ \mathbf{rev} \qquad \Delta\langle \overrightarrow{P} : z\rangle \Rightarrow D : \omega \quad \neg\, \mathbf{foc} \wedge ?P\ \mathbf{rev}}{\Delta\langle \Gamma \mid_k \overrightarrow{\boxed{Q \downarrow_k P}} : y\rangle \Rightarrow D : \omega\{(y\ \phi)/z\} \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}} \downarrow_k L$$

$$\frac{\Gamma \Rightarrow \boxed{P} : \psi \quad \mathbf{foc} \wedge \neg\, \mathbf{rev} \qquad \Delta\langle \overrightarrow{\boxed{Q}} : z\rangle \Rightarrow D : \omega \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}}{\Delta\langle \overrightarrow{\boxed{Q \uparrow_k P}} : x \mid_k \Gamma\rangle \Rightarrow D : \omega\{(x\ \psi)/z\} \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}} \uparrow_k L$$

$$\frac{\Gamma \Rightarrow Q_1 : \psi \quad \neg\, \mathbf{foc} \wedge ?Q_1\ \mathbf{rev} \qquad \Delta\langle \overrightarrow{\boxed{Q_2}} : z\rangle \Rightarrow D : \omega \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}}{\Delta\langle \overrightarrow{\boxed{Q_2 \uparrow_k Q_1}} : x \mid_k \Gamma\rangle \Rightarrow D : \omega\{(x\ \psi)/z\} \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}} \uparrow_k L$$

$$\frac{\Gamma \Rightarrow \boxed{P_1} : \psi \quad \mathbf{foc} \wedge \neg\, \mathbf{rev} \qquad \Delta\langle \overrightarrow{P_2} : z\rangle \Rightarrow D : \omega \quad \neg\, \mathbf{foc} \wedge ?P_2\ \mathbf{rev}}{\Delta\langle \overrightarrow{\boxed{P_2 \uparrow_k P_1}} : x \mid_k \Gamma\rangle \Rightarrow D : \omega\{(x\ \psi)/z\} \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}} \uparrow_k L$$

$$\frac{\Gamma \Rightarrow Q : \psi \quad \neg\, \mathbf{foc} \wedge ?Q\ \mathbf{rev} \qquad \Delta\langle \overrightarrow{P} : z\rangle \Rightarrow D : \omega \quad \neg\, \mathbf{foc} \wedge ?P\ \mathbf{rev}}{\Delta\langle \overrightarrow{\boxed{P \uparrow_k Q}} : x \mid_k \Gamma\rangle \Rightarrow D : \omega\{(x\ \psi)/z\} \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}} \uparrow_k L$$

Figure 10: Left irreversible discontinuous multiplicative rules

$$\frac{\Delta \Rightarrow \boxed{P_1} : \phi \quad \mathbf{foc} \wedge \neg\, \mathbf{rev} \qquad \Gamma \Rightarrow \boxed{P_2} : \psi \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}}{\Delta, \Gamma \Rightarrow \boxed{P_1 \bullet P_2} : (\phi, \psi) \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}} \bullet R$$

$$\frac{\Delta \Rightarrow \boxed{P} : \phi \quad \mathbf{foc} \wedge \neg\, \mathbf{rev} \qquad \Gamma \Rightarrow Q : \psi \quad \neg\, \mathbf{foc} \wedge ?Q\ \mathbf{rev}}{\Delta, \Gamma \Rightarrow \boxed{P \bullet Q} : (\phi, \psi) \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}} \bullet R$$

$$\frac{\Delta \Rightarrow N : \phi \quad \neg\, \mathbf{foc} \wedge ?N\ \mathbf{rev} \qquad \Gamma \Rightarrow \boxed{P} : \psi \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}}{\Delta, \Gamma \Rightarrow \boxed{N \bullet P} : (\phi, \psi) \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}} \bullet R$$

$$\frac{\Delta \Rightarrow N_1 : \phi \neg\, \mathbf{foc} \wedge ?N_1\ \mathbf{rev} \qquad \Gamma \Rightarrow N_2 : \psi \quad \mathbf{foc} \wedge ?N_2\ \mathbf{rev}}{\Delta, \Gamma \Rightarrow \boxed{N_1 \bullet N_2} : (\phi, \psi) \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}} \bullet R$$

$$\frac{}{\Lambda \Rightarrow \boxed{I} : 0\ \mathbf{foc} \wedge \neg\, \mathbf{rev}} IR$$

Figure 11: Right irreversible continuous multiplicative rules

$$\frac{\Delta \Rightarrow \boxed{P_1} : \phi \quad \mathbf{foc} \wedge \neg\, \mathbf{rev} \qquad \Gamma \Rightarrow \boxed{P_2} : \psi \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}}{\Delta \mid_k \Gamma \Rightarrow \boxed{P_1 \odot_k P_2} : (\phi, \psi) \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}} \odot_k R$$

$$\frac{\Delta \Rightarrow \boxed{P} : \phi \quad \mathbf{foc} \wedge \neg\, \mathbf{rev} \qquad \Gamma \Rightarrow Q : \psi \quad \neg\, \mathbf{foc} \wedge ?N\ \mathbf{rev}}{\Delta \mid_k \Gamma \Rightarrow \boxed{P \odot_k Q} : (\phi, \psi) \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}} \odot_k R$$

$$\frac{\Delta \Rightarrow Q : \phi \quad \neg\, \mathbf{foc} \wedge ?N\ \mathbf{rev} \qquad \Gamma \Rightarrow \boxed{P} : \psi \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}}{\Delta \mid_k \Gamma \Rightarrow \boxed{Q \odot_k P} : (\phi, \psi) \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}} \odot_k R$$

$$\frac{\Delta \Rightarrow Q_1 : \phi \quad \neg\, \mathbf{foc} \wedge ?Q_1\ \mathbf{rev} \qquad \Gamma \Rightarrow Q_2 : \psi \quad \neg\, \mathbf{foc} \wedge ?Q_2\ \mathbf{rev}}{\Delta \mid_k \Gamma \Rightarrow \boxed{Q_1 \odot_k Q_2} : (\phi, \psi) \quad \mathbf{foc} \wedge \neg\, \mathbf{rev}} \odot_k R$$

$$\frac{}{1 \Rightarrow \boxed{J} : 0\ \mathbf{foc} \wedge \neg\, \mathbf{rev}} JR$$

Figure 12: Right irreversible discontinuous multiplicative rules

4 Count-invariance

We define infinitary count invariance for categorial logic extending count invariance for multiplicatives (van Benthem 1991[51]) and additives and bracket modalities (Valentín et al. 2013[50]) to include exponentials. This affords effective pruning of proof search in categorial parsing/theorem-proving.

Count invariance for multiplicatives in (sub)linear logic is introduced in van Benthem (1991[51]). This involves simply checking the number of positive and negative occurrences of each atom in a sequent. Thus where $\#(\Sigma)$ is a count of the sequent Σ we have:

(5) $\vdash \Sigma \Longrightarrow \#(\Sigma) = 0$

I.e. the numbers of positive and negative occurrences of each atom must exactly balance. This provides a necessary, but of course not sufficient, criterion for theoremhood, and it can be checked rapidly. It can be used as a filter in proof search: if backward chaining proof search generates a goal which does not satisfy the count invariant, the goal can be safely made to fail immediately. This notion of count for multiplicatives was included in the categorial parser/theorem-prover CatLog1 (Morrill 2012[33]).

In Valentín et al. (2013[50]) the idea is extended to additives (and bracket modalities). Instead of a single count for each atom of a sequent Σ we have a minimum count $\#_{\min}(\Sigma)$ and a maximum count $\#_{\max}(\Sigma)$ and for a sequent to be a theorem it must satisfy two inequations:

(6) $\vdash \Sigma \Longrightarrow \#_{\min}(\Sigma) \leq 0 \leq \#_{\max}(\Sigma)$

I.e. the count functions $\#_{\min}$ and $\#_{\max}$ define an interval which must include the point of balance 0; for the multiplicatives, $\#_{\min} = \#_{\max} = \#$ and (6) reduces to the special case (5). This count-invariance is included in the categorial parser/theorem-prover CatLog2. Here we describe the count-invariance of CatLog3 which includes further infinitary count functions for exponentials (Kuznetsov et al. 2017[16]).

We consider terms built over the constants 0, 1, $\bot$ (minus infinity, $-\infty$), and $\top$ (plus infinity, $+\infty$) by operations plus (+), minus (−), minimum (min) and maximum (max), and infinitary step functions X and Y thus; $i, j \in \mathcal{Z}$ and $n \in \mathcal{Z}^+$:

$+$	j	$\bot$	$\top$
i	$i+j$	$\bot$	$\top$
$\bot$	$\bot$	$\bot$	$*$
$\top$	$\top$	$*$	$\top$

$-$	j	$\bot$	$\top$
i	$i-j$	$\top$	$\bot$
$\bot$	$\bot$	$*$	$\bot$
$\top$	$\top$	$\top$	$*$

min	j	$\bot$	$\top$
i	$\frac{\lvert i+j\rvert-\lvert i-j\rvert}{2}$	$\bot$	i
$\bot$	$\bot$	$\bot$	$\bot$
$\top$	j	$\bot$	$\top$

max	j	$\bot$	$\top$
i	$\frac{\lvert i+j\rvert+\lvert i-j\rvert}{2}$	i	$\top$
$\bot$	j	$\bot$	$\top$
$\top$	$\top$	$\top$	$\top$

$$X(i) = \begin{cases} \top & \text{if } i > 0 \\ i & \text{if } i \leq 0 \end{cases} \qquad Y(i) = \begin{cases} i & \text{if } i \geq 0 \\ \bot & \text{if } i < 0 \end{cases}$$

Where $\mathcal{P}$ is the set of primitive types, $P \in \mathcal{P}$, $Q \in \mathcal{P} \cup \{[]\}$, $p \in \{^\bullet, ^\circ\}$, and $\overline{^\bullet} = ^\circ$ and $\overline{^\circ} = ^\bullet$ we define the count functions for **DA1S4b!$_\mathbf{b}$?** as shown in Figure 13.

For zones, stoups, tree terms and configurations, counts are as follows:

$$\begin{array}{rcl}
\#_{m,Q}(\mathcal{S};\mathcal{O}) & = & \#_{m,Q}(\mathcal{S}) + \#_{m,Q}(\mathcal{O}) \\
\#_{m,Q}(\emptyset) & = & 0 \\
\#_{m,Q}(\mathcal{F},\mathcal{S}) & = & \#_{m,Q}(\mathcal{F}) + \#_{m,Q}(\mathcal{S}) \\
\#_{m,Q}(\Lambda) & = & 0 \\
\#_{m,Q}(\mathcal{T},\mathcal{O}) & = & \#_{m,Q}(\mathcal{T}) + \#_{m,Q}(\mathcal{O}) \\
\#_{m,Q}(1) & = & 0 \\
\#_{m,Q}(\mathcal{F}) & = & \#^\bullet_{m,Q}(\mathcal{F}) \\
\#_{m,Q}(\mathcal{F}\{\mathcal{O}_1 : \ldots : \mathcal{O}_i\}) & = & \#^\bullet_{m,Q}(\mathcal{F}) + \Sigma^i_{n=1}\#_{m,Q}(\mathcal{O}_n) \\
\#_{m,[]}([\mathcal{Z}]) & = & \#_{m,[]}(\mathcal{Z}) + 1 \\
\#_{m,P}([\mathcal{Z}]) & = & \#_{m,P}(\mathcal{Z})
\end{array}$$

The count-invariance theorem is:

(7) **Theorem**.

$\vdash \Xi \Rightarrow A \Longrightarrow \forall Q \in \mathcal{P} \cup \{[]\}, \#_{\min,Q}(\Xi \Rightarrow A) \leq 0 \leq \#_{\max,Q}(\Xi \Rightarrow A)$

where, $\#_{m,Q}(\Xi \Rightarrow A) = \#^\circ_{m,Q}(A) - \#_{\overline{m},Q}(\Xi)$.

Relativisation including medial and parasitic extraction is obtained by assigning a relative pronoun a type $(CN\backslash CN)/(!N\backslash S)$ whereby the body of a relative clause is analysed as $!N\backslash S$. By way of example of count-invariance, we show how it discards $N, N\backslash S \Rightarrow !N\backslash S$ corresponding to the ungrammaticality of a relative clause without a gap: **paper that John walks*. We have the max N-count: $\#_{\max,N}(N, N\backslash S \Rightarrow !N\backslash S) = \#^\circ_{\max,N}(!N\backslash S) - \#^\bullet_{\min,N}(N, N\backslash S) = \#^\circ_{\max,N}(S) - \#^\bullet_{\min,N}(!N) - \#^\bullet_{\min,N}(N) - \#^\bullet_{\min,N}(N\backslash S) = 0 - Y(\#^\bullet_{\min,N}(N)) - 1 - \#^\bullet_{\min,N}(S) + \#^\circ_{\min,N}(N) = -Y(1) - 1 - 0 + 1 = -1 - 1 + 1 = -1 \not\geq 0$ which means that the count-invariance is not satisfied.

Iterated sentential coordination is obtained by assigning a coordinator the type $(?S\backslash S)/S$. By way of a second example we show how count-invariance discards $N, N, N\backslash S \Rightarrow ?S$ corresponding to the ungrammaticality of unequilibrated coordination: **John Mary walks and Suzy talks*. For this example maximum N-count is: $\#_{\max,N}(N, N, N\backslash S \Rightarrow ?S) = \#^\circ_{\max,N}(?S) - \#^\bullet_{\min,N}(N, N, N\backslash S) = X(\#^\circ_{\max,N}(S)) - \#^\bullet_{\min,N}(N) - \#^\bullet_{\min,N}(N) - \#^\bullet_{\min,N}(N\backslash S) = X(0) - 1 - 1 - \#^\bullet_{\min,N}(S) + \#^\bullet_{\max,N}(N) = 0 - 2 - 0 + 1 = -1 \not\geq 0$ which means that the count-invariance is not satisfied.

$$
\begin{array}{rcl}
\#^p_{m,Q}(P) & = & \begin{array}{ll} 1 & \text{if } Q = P \\ 0 & \text{if } Q \neq P \end{array} \\
\#^p_{m,Q}(A\backslash C) & = & \#^p_{m,Q}(A{\downarrow_k}C) \quad = \quad \#^p_{m,Q}(C) - \#^{\overline{p}}_{\overline{m},Q}(A) \\
\#^p_{m,Q}(C/B) & = & \#^p_{m,Q}(C{\uparrow_k}B) \quad = \quad \#^p_{m,Q}(C) - \#^{\overline{p}}_{\overline{m},Q}(B) \\
\#^p_{m,Q}(A \bullet B) & = & \#^p_{m,Q}(A \odot_k B) \quad = \quad \#^p_{m,Q}(A) + \#^p_{m,Q}(B) \\
\#^p_{m,Q}(I) & = & \#^p_{m,Q}(J) \quad = \quad 0 \\
\#^{\circ}_{m,Q}(A \& B) & = & \overline{m}(\#^{\circ}_{m,Q}(A), \#^{\circ}_{m,Q}(B)) \\
\#^{\bullet}_{m,Q}(A \& B) & = & m(\#^{\bullet}_{m,Q}(A), \#^{\bullet}_{m,Q}(B)) \\
\#^{\circ}_{m,Q}(A \oplus B) & = & m(\#^{\circ}_{m,Q}(A), \#^{\circ}_{m,Q}(B)) \\
\#^{\bullet}_{m,Q}(A \oplus B) & = & \overline{m}(\#^{\bullet}_{m,Q}(A), \#^{\bullet}_{m,Q}(B)) \\
\#^p_{m,Q}(\bigwedge xA) & = & \#^p_{m,Q}(\bigvee xA) \quad = \quad \#^p_{m,Q}(A) \\
\#^p_{m,Q}(\Box A) & = & \#^p_{m,Q}(\Diamond A) \quad = \quad \#^p_{m,Q}(A) \\
\#^p_{m,P}([\,]^{-1}A) & = & \#^p_{m,P}(A) \\
\#^p_{m,[]}([\,]^{-1}A) & = & \#^p_{m,[]}(A) - 1 \\
\#^p_{m,P}(\langle\rangle A) & = & \#^p_{m,P}(A) \\
\#^p_{m,[]}(\langle\rangle A) & = & \#^p_{m,[]}(A) + 1 \\
\#^{\bullet}_{\min,Q}(!A) & = & Y(\#^{\bullet}_{\min,Q}(A)) \\
\#^{\bullet}_{\max,P}(!A) & = & X(\#^{\bullet}_{\max,P}(A)) \\
\#^{\bullet}_{\max,[]}(!A) & = & \top \\
\#^{\circ}_{m,Q}(!A) & = & \#^{\circ}_{m,Q}(A) \\
\#^{\circ}_{\max,Q}(?A) & = & X(\#^{\circ}_{\max,Q}(A)) \\
\#^{\circ}_{\min,Q}(?A) & = & Y(\#^{\circ}_{\min,Q}(A)) \\
\#^{\bullet}_{m,Q}(?A) & = & \#^{\bullet}_{m,Q}(A)
\end{array}
$$

Figure 13: Count function

```
str(dwp('(7-7)'), [b([john]), walks], s(f)).
str(dwp('(7-16)'), [b([every, man]), talks], s(f)).
str(dwp('(7-19)'), [b([the, fish]), walks], s(f)).
str(dwp('(7-32)'), [b([every, man]), b([b([walks, or, talks])])], s(f)).
str(dwp('(7-34)'), [b([b([b([every, man]), walks, or, b([every, man]), talks])])], s(f)).
str(dwp('(7-39)'), [b([b([b([a, woman]), walks, and, b([she]), talks])])], s(f)).
str(dwp('(7-43, 45)'), [b([john]), believes, that, b([a, fish]), walks], s(f)).
str(dwp('(7-48, 49, 52)'), [b([every, man]), believes, that, b([a, fish]), walks], s(f)).
str(dwp('(7-57)'), [b([every, fish, such, that, b([it]), walks]), talks], s(f)).
str(dwp('(7-60, 62)'), [b([john]), seeks, a, unicorn], s(f)).
str(dwp('(7-73)'), [b([john]), is, bill], s(f)).
str(dwp('(7-76)'), [b([john]), is, a, man], s(f)).
str(dwp('(7-83)'), [necessarily, b([john]), walks], s(f)).
str(dwp('(7-86)'), [b([john]), walks, slowly], s(f)).
str(dwp('(7-91)'), [b([john]), tries, to, walk], s(f)).
str(dwp('(7-94)'), [b([john]), tries, to, b([b([catch, a, fish, and, eat, it])])], s(f)).
str(dwp('(7-98)'), [b([john]), finds, a, unicorn], s(f)).
str(dwp('(7-105)'), [b([every, man, such, that, b([he]), loves, a, woman]), loses, her], s(f)).
str(dwp('(7-110)'), [b([john]), walks, in, a, park], s(f)).
str(dwp('(7-116, 118)'), [b([every, man]), doesnt, walk], s(f)).
```

Figure 14: Montague sentences

5 Illustration

Morrill and Valentín (2016[43]) defines as the *Montague Test* the task of providing a computational grammar of the PTQ fragment of Montague (1973[21]), and shows how CatLog fulfils this task. We are not aware of any other system which has passed the Montague Test. The example sentences of Chapter 7 of Dowty et al. (1981[7]) are given in Figure 14; the lexicon is given in Figure 15.

The CatLog3 LaTeX output for the (ambiguous) last sentence is as follows:

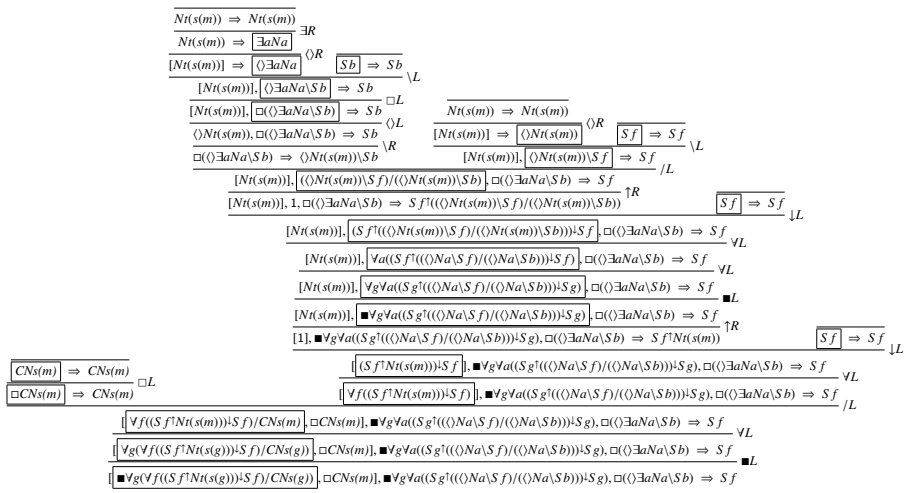
(dwp((7-116, 118))) [**every**+**man**]+**doesnt**+**walk** : $S\,f$

$[\blacksquare\forall g(\forall f((S\,f{\uparrow}Nt(s(g))){\downarrow}S\,f)/CNs(g)) : \lambda A\lambda B\forall C[(A\ C) \rightarrow (B\ C)],$
$\square CNs(m) : man], \blacksquare\forall g\forall a((S\,g{\uparrow}((\langle\rangle Na\backslash S\,f)/(\langle\rangle Na\backslash S\,b))){\downarrow}S\,g) : \lambda D\neg(D\ \lambda E\lambda F(E\ F)),$
$\square(\langle\rangle(\exists aNa{-}\exists gNt(s(g)))\backslash S\,f) : \hat{}\lambda G(Pres\ (\check{}walk\ G))\ \Rightarrow\ S\,f$

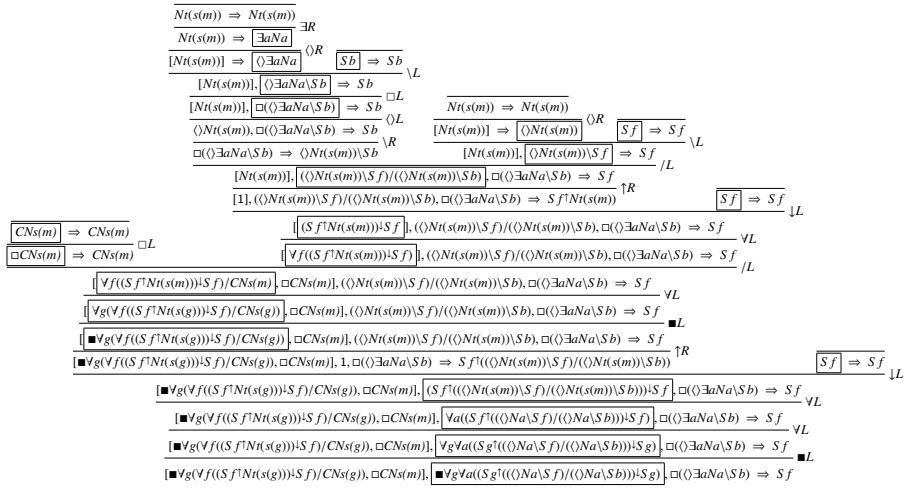
$[\blacksquare\forall g(\forall f((S\,f{\uparrow}Nt(s(g))){\downarrow}S\,f)/CNs(g)) : \lambda A\lambda B\forall C[(A\ C) \rightarrow (B\ C)],$
$\square CNs(m) : man], \blacksquare\forall g\forall a((S\,g{\uparrow}((\langle\rangle Na\backslash S\,f)/(\langle\rangle Na\backslash S\,b))){\downarrow}S\,g) : \lambda D\neg(D\ \lambda E\lambda F(E\ F)),$
$\square(\langle\rangle\exists aNa\backslash S\,b) : \hat{}\lambda G(\check{}walk\ G)\ \Rightarrow\ S\,f$

a : $\blacksquare\forall g(\forall f((S\,f\uparrow\blacksquare Nt(s(g)))\downarrow S\,f)/CNs(g)) : \lambda A\lambda B\exists C[(A\ C)\wedge(B\ C)]$
and : $\blacksquare\forall f((\blacksquare ?S\,f\backslash[]^{-1}[]^{-1}S\,f)/\blacksquare S\,f) : (\Phi^{n+}\ 0\ and)$
and : $\blacksquare\forall a\forall f((\blacksquare ?(\langle\rangle Na\backslash S\,f)\backslash[]^{-1}[]^{-1}(\langle\rangle Na\backslash S\,f))/\blacksquare(\langle\rangle Na\backslash S\,f)) : (\Phi^{n+}\ (s\ 0)\ and)$
believes : $\square(((\langle\rangle\exists gNt(s(g))\backslash S\,f)/(CPthat\sqcup\square S\,f)) : \hat{}\lambda A\lambda B(Pres\ ((\check{}believe\ A)\ B))$
bill : $\blacksquare Nt(s(m)) : b$
catch : $\square(((\langle\rangle\exists aNa\backslash S\,b)/\exists aNa) : \hat{}\lambda A\lambda B((\check{}catch\ A)\ B)$
doesnt : $\blacksquare\forall g\forall a((S\,g\uparrow(((\langle\rangle Na\backslash S\,f)/(\langle\rangle Na\backslash S\,b)))\downarrow S\,g) : \lambda A\neg(A\ \lambda B\lambda C(B\ C))$
eat : $\square(((\langle\rangle\exists aNa\backslash S\,b)/\exists aNa) : \hat{}\lambda A\lambda B((\check{}eat\ A)\ B)$
every : $\blacksquare\forall g(\forall f((S\,f\uparrow Nt(s(g)))\downarrow S\,f)/CNs(g)) : \lambda A\lambda B\forall C[(A\ C)\rightarrow(B\ C)]$
finds : $\square(((\langle\rangle\exists gNt(s(g))\backslash S\,f)/\exists aNa) : \hat{}\lambda A\lambda B(Pres\ ((\check{}find\ A)\ B))$
fish : $\square CNs(n) : fish$
he : $\blacksquare[]^{-1}\forall g((\blacksquare S\,g|\blacksquare Nt(s(m)))/(\langle\rangle Nt(s(m))\backslash S\,g)) : \lambda AA$
her : $\blacksquare\forall g\forall a((((\langle\rangle Na\backslash S\,g)\uparrow\blacksquare Nt(s(f)))\downarrow(\blacksquare(\langle\rangle Na\backslash S\,g)|\blacksquare Nt(s(f)))) : \lambda AA$
in : $\square(\forall a\forall f(((\langle\rangle Na\backslash S\,f)\backslash(\langle\rangle Na\backslash S\,f))/\exists aNa) : \hat{}\lambda A\lambda B\lambda C((\check{}in\ A)\ (B\ C))$
is : $\blacksquare(((\langle\rangle\exists gNt(s(g))\backslash S\,f)/(\exists aNa\oplus(\exists g((CNg/CNg)\sqcup(CNg\backslash CNg))-I))) :$
$\lambda A\lambda B(Pres\ (A\rightarrow C.[B=C];D.((D\ \lambda E[E=B])\ B)))$
it : $\blacksquare\forall f\forall a((((\langle\rangle Na\backslash S\,f)\uparrow\blacksquare Nt(s(n)))\downarrow(\blacksquare(\langle\rangle Na\backslash S\,f)|\blacksquare Nt(s(n)))) : \lambda AA$
it : $\blacksquare[]^{-1}\forall f((\blacksquare S\,f|\blacksquare Nt(s(n)))/(\langle\rangle Nt(s(n))\backslash S\,f)) : \lambda AA$
john : $\blacksquare Nt(s(m)) : j$
loses : $\square(((\langle\rangle\exists gNt(s(g))\backslash S\,f)/\exists aNa) : \hat{}\lambda A\lambda B(Pres\ ((\check{}lose\ A)\ B))$
loves : $\square(((\langle\rangle\exists gNt(s(g))\backslash S\,f)/\exists aNa) : \hat{}\lambda A\lambda B(Pres\ ((\check{}love\ A)\ B))$
man : $\square CNs(m) : man$
necessarily : $\blacksquare(S\,A/\square S\,A) : Nec$
or : $\blacksquare\forall f((\blacksquare ?S\,f\backslash[]^{-1}[]^{-1}S\,f)/\blacksquare S\,f) : (\Phi^{n+}\ 0\ or)$
or : $\blacksquare\forall a\forall f((\blacksquare ?(\langle\rangle Na\backslash S\,f)\backslash[]^{-1}[]^{-1}(\langle\rangle Na\backslash S\,f))/\blacksquare(\langle\rangle Na\backslash S\,f)) : (\Phi^{n+}\ (s\ 0)\ or)$
or : $\blacksquare\forall f((\blacksquare ?(S\,f/(\langle\rangle\exists gNt(s(g))\backslash S\,f))\backslash[]^{-1}[]^{-1}(S\,f/(\langle\rangle\exists gNt(s(g))\backslash S\,f)))/\blacksquare(S\,f/(\langle\rangle\exists gNt(s(g))\backslash S\,f))) :$
$(\Phi^{n+}\ (s\ 0)\ or)$
park : $\square CNs(n) : park$
seeks : $\square(((\langle\rangle\exists gNt(s(g))\backslash S\,f)/\square\forall a\forall f(((Na\backslash S\,f)/\exists bNb)\backslash(Na\backslash S\,f))) :$
$\hat{}\lambda A\lambda B((\check{}tries\ \hat{}((\check{}A\ \check{}find)\ B))\ B)$
she : $\blacksquare[]^{-1}\forall g((\blacksquare S\,g|\blacksquare Nt(s(f)))/(\langle\rangle Nt(s(f))\backslash S\,g)) : \lambda AA$
slowly : $\square\forall a\forall f(\square(\langle\rangle Na\backslash S\,f)\backslash(\langle\rangle\square Na\backslash S\,f)) : \hat{}\lambda A\lambda B(\check{}slowly\ \hat{}(\check{}A\ \check{}B))$
such+that : $\blacksquare\forall n((CNn\backslash CNn)/(S\,f|\blacksquare Nt(n))) : \lambda A\lambda B\lambda C[(B\ C)\wedge(A\ C)]$
talks : $\square(\langle\rangle\exists gNt(s(g))\backslash S\,f) : \hat{}\lambda A(Pres\ (\check{}talk\ A))$
that : $\blacksquare(CPthat/\square S\,f) : \lambda AA$
the : $\blacksquare\forall n(Nt(n)/CNn) : \iota$
to : $\blacksquare((PPto/\exists aNa)\sqcap\forall n((\langle\rangle Nn\backslash S\,i)/(\langle\rangle Nn\backslash S\,b))) : \lambda AA$
tries : $\square(((\langle\rangle\exists gNt(s(g))\backslash S\,f)/\square(\langle\rangle\exists gNt(s(g))\backslash S\,i)) : \hat{}\lambda A\lambda B((\check{}tries\ \hat{}(\check{}A\ B))\ B)$
unicorn : $\square CNs(n) : unicorn$
walk : $\square(\langle\rangle\exists aNa\backslash S\,b) : \hat{}\lambda A(\check{}walk\ A)$
walks : $\square(\langle\rangle\exists gNt(s(g))\backslash S\,f) : \hat{}\lambda A(Pres\ (\check{}walk\ A))$
woman : $\square CNs(f) : woman$

Figure 15: Montague lexicon



$\forall C[(\check{}man\ C) \rightarrow \neg(\check{}walk\ C)]$



$\neg\forall G[(\check{}man\ G) \rightarrow (\check{}walk\ G)]$

References

[1] Kazimierz Ajdukiewicz. Die syntaktische Konnexität. *Studia Philosophica*, 1:1–27, 1935. Translated in Storrs McCall, editor, 1967, *Polish Logic: 1920–1939*, Oxford University Press, Oxford, 207–231.

[2] J. M. Andreoli. Logic programming with focusing in linear logic. *Journal of Logic and Computation*, 2(3):297–347, 1992.

[3] Yehoshua Bar-Hillel. A quasi-arithmetical notation for syntactic description. *Language*, 29:47–58, 1953.

[4] Guy Barry, Mark Hepple, Neil Leslie, and Glyn Morrill. Proof Figures and Structural Operators for Categorial Grammar. In *Proceedings of the Fifth Conference of the European Chapter of the Association for Computational Linguistics*, pages 198–203, Berlin, 1991.

[5] Bob Carpenter. *Type-Logical Semantics*. MIT Press, Cambridge, MA, 1997.

[6] Noam Chomsky. *Syntactic Structures*. Mouton, The Hague, 1957.

[7] David R. Dowty, Robert E. Wall, and Stanley Peters. *Introduction to Montague Semantics*, volume 11 of *Synthese Language Library*. D. Reidel, Dordrecht, 1981.

[8] Mario Fadda. *Geometry of Grammar: Exercises in Lambek Style*. PhD thesis, Universitat Politècnica de Catalunya, Barcelona, 2010.

[9] G. Gentzen. Untersuchungen über das logische Schliessen. *Mathematische Zeitschrift*, 39:176–210 and 405–431, 1934. Translated in M.E. Szabo, editor, 1969, *The Collected Papers of Gerhard Gentzen*, North-Holland, Amsterdam, 68–131.

[10] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[11] Jean-Yves Girard. *The Blind Spot*. European Mathematical Society, Zürich, 2011.

[12] Mark Hepple. *The Grammar and Processing of Order and Dependency*. PhD thesis, University of Edinburgh, 1990.

[13] Gerhard Jäger. *Anaphora and Type Logical Grammar*, volume 24 of *Trends in Logic – Studia Logica Library*. Springer, Dordrecht, 2005.

[14] M. Kanazawa. The Lambek calculus enriched with additional connectives. *Journal of Logic, Language and Information*, 1:141–171, 1992.

[15] Max Kanovich, Stepan Kuznetsov, and Andre Scedrov. Undecidability of the Lambek calculus with a relevant modality. In Annie Foret, Glyn Morrill, Reinhard Muskens, Rainer Osswald, and Sylvain Pogodalla, editors, *Formal Grammar 2015: Revised Selected Papers. Formal Grammar 2016: Proceedings*, volume 9804, pages 240–246, Berlin, 2016. Springer.

[16] Stepan Kuznetsov, Glyn Morrill, and Oriol Valentín. Count-invariance including exponentials. In Makoto Kanazawa, editor, *Mathematics of Language*, London, 2017.

[17] J. Lambek. On the Calculus of Syntactic Types. In Roman Jakobson, editor, *Structure of Language and its Mathematical Aspects, Proceedings of the Symposia in Applied Mathematics XII*, pages 166–178. American Mathematical Society, Providence, Rhode Island, 1961.

[18] J. Lambek. Categorial and Categorical Grammars. In Richard T. Oehrle, Emmon Bach, and Deidre Wheeler, editors, *Categorial Grammars and Natural Language Structures*, volume 32 of *Studies in Linguistics and Philosophy*, pages 297–317. D. Reidel, Dordrecht, 1988.

[19] Joachim Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, 65:154–170, 1958.

[20] Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51(1-2):125–157, 1991.

[21] Richard Montague. The Proper Treatment of Quantification in Ordinary English. In J. Hintikka, J.M.E. Moravcsik, and P. Suppes, editors, *Approaches to Natural Language: Proceedings of the 1970 Stanford Workshop on Grammar and Semantics*, pages 189–224. D. Reidel, Dordrecht, 1973. Reprinted in R.H. Thomason, editor, 1974, *Formal Philosophy: Selected Papers of Richard Montague*, Yale University Press, New Haven, 247–270.

[22] Michael Moortgat. *Categorial Investigations: Logical and Linguistic Aspects of the Lambek Calculus*. Foris, Dordrecht, 1988. PhD thesis, Universiteit van Amsterdam.

[23] Michael Moortgat. Multimodal linguistic inference. *Journal of Logic, Language and Information*, 5(3, 4):349–385, 1996. Also in *Bulletin of the IGPL*, 3(2,3):371–401, 1995.

[24] Michael Moortgat. Categorial Type Logics. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, pages 93–177. Elsevier Science B.V. and the MIT Press, Amsterdam and Cambridge, Massachusetts, 1997.

[25] Richard Moot. Extended Lambek Calculi and First-Order Linear Logic. In Michael Moortgat Claudia Casadio, Bob Coeke and Philip Scott, editors, *Categories and Types in Logic, Language and Physics: Essays Dedicated to Jim Lambek on the Occasion of His 90th Birthday*, volume 8222 of *LNCS, FoLLI Publications in Logic, Language and Information*, pages 297–330. Springer, Berlin, 2014.

[26] Richard Moot. Proof nets for the displacement calculus. In Annie Foret, Glyn Morrill, Reinhard Muskens, Rainer Osswald, and Sylvain Pogodalla, editors, *Formal Grammar: 20th and 21st International Conferences*, volume 9804 of

LNCS, FoLLI Publications in Logic, Language and Information, pages 273–289, Berlin, 2016. Springer.

[27] Richard Moot and Christian Retoré. *The Logic of Categorial Grammars: A Deductive Account of Natural Language Syntax and Semantics*. Springer, Heidelberg, 2012.

[28] Glyn Morrill. Grammar and Logical Types. In Martin Stockhof and Leen Torenvliet, editors, *Proceedings of the Seventh Amsterdam Colloquium*, pages 429–450, Amsterdam, 1990. Universiteit van Amsterdam.

[29] Glyn Morrill. Intensionality and Boundedness. *Linguistics and Philosophy*, 13(6):699–726, 1990.

[30] Glyn Morrill. Categorial Formalisation of Relativisation: Pied Piping, Islands, and Extraction Sites. Technical Report LSI-92-23-R, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, 1992.

[31] Glyn Morrill. Categorial grammar. In Bernd Heine and Heiko Narrog, editors, *The Oxford Handbook of Linguistic Analysis*, pages 67–86. Oxford University Press, Oxford, 2010.

[32] Glyn Morrill. Logic Programming of the Displacement Calculus. In Sylvain Pogodalla and Jean-Philippe Prost, editors, *Proceedings of Logical Aspects of Computational Linguistics 2011, LACL'11, Montpellier*, number LNAI 6736 in Springer Lecture Notes in AI, pages 175–189, Berlin, 2011. Springer.

[33] Glyn Morrill. CatLog: A Categorial Parser/Theorem-Prover. In *LACL 2012 System Demonstrations*, Logical Aspects of Computational Linguistics 2012, pages 13–16, Nantes, 2012.

[34] Glyn Morrill. Logical grammar. In Ruth Kempson, Tim Fernando, and Nicholas Asher, editors, *The Handbook of Philosophy of Linguistics*, pages 63–92. Elsevier, Oxford and Amsterdam, 2012.

[35] Glyn Morrill. Grammar logicised: relativisation. *Linguistics and Philosophy*, 40(2):119–163, 2017.

[36] Glyn Morrill and Mario Fadda. Proof Nets for Basic Discontinuous Lambek Calculus. *Logic and Computation*, 18(2):239–256, 2008.

[37] Glyn Morrill and Josep-Maria Merenciano. Generalising Discontinuity. *Traitement automatique des langues*, 37(2):119–143, 1996.

[38] Glyn Morrill and Oriol Valentín. Displacement Calculus. *Linguistic Analysis*, 36(1–4):167–192, 2010. Special issue Festschrift for Joachim Lambek.

[39] Glyn Morrill and Oriol Valentín. Displacement Logic for Anaphora. *Journal of Computing and System Science*, 80:390–409, 2014. http://dx.doi.org/10.1016/j.jcss.2013.05.006.

[40] Glyn Morrill and Oriol Valentín. Semantically Inactive Multiplicatives and Words as Types. In Nicholas Asher and Sergei Soloviev, editors, *Proceedings of Logical Aspects of Computational Linguistics, LACL'14, Toulouse*, number 8535 in LNCS, FoLLI Publications on Logic, Language and Information, pages 149–162, Berlin, 2014. Springer.

[41] Glyn Morrill and Oriol Valentín. Computational Coverage of TLG: Nonlinearity. In M. Kanazawa, L.S. Moss, and V. de Paiva, editors, *Proceedings of NLCS'15. Third Workshop on Natural Language and Computer Science*, volume 32 of *EPiC*, pages 51–63, Kyoto, 2015. Workshop affiliated with Automata, Languages and Programming (ICALP) and Logic in Computer Science (LICS).

[42] Glyn Morrill and Oriol Valentín. Multiplicative-Additive Focusing for Parsing as Deduction. In I. Cervesato and C. Schürmann, editors, *First International Workshop on Focusing, workshop affiliated with LPAR 2015*, number 197 in EPTCS, pages 29–54, Suva, Fiji, 2015.

[43] Glyn Morrill and Oriol Valentín. Computational coverage of Type Logical Grammar: The Montague Test. In C. Piñón, editor, *Empirical Issues in Syntax and Semantics*, volume 11, pages 141–170. Colloque de Syntaxe et Sémantique à Paris (CSSP), Paris, 2016.

[44] Glyn Morrill and Oriol Valentín. On the Logic of Expansion in Natural Language. In Maxime Amblard, Phillipe de Groote, Sylvain Pogodalla, and Christian Retoré, editors, *Proceedings of Logical Aspects of Computational Linguistics, LACL'16, Nancy*, volume 10054 of *LNCS, FoLLI Publications on Logic, Language and Information*, pages 228–246, Berlin, 2016. Springer.

[45] Glyn Morrill, Oriol Valentín, and Mario Fadda. Dutch Grammar and Processing: A Case Study in TLG. In Peter Bosch, David Gabelaia, and Jérôme Lang, editors, *Logic, Language, and Computation: 7th International Tbilisi Symposium, Revised Selected Papers*, number 5422 in Lecture Notes in Artificial Intelligence, pages 272–286, Berlin, 2009. Springer.

[46] Glyn Morrill, Oriol Valentín, and Mario Fadda. The Displacement Calculus. *Journal of Logic, Language and Information*, 20(1):1–48, 2011.

[47] Glyn V. Morrill. *Type Logical Grammar: Categorial Logic of Signs*. Kluwer Academic Publishers, Dordrecht, 1994.

[48] Glyn V. Morrill. *Categorial Grammar: Logical Syntax, Semantics, and Processing*. Oxford University Press, New York and Oxford, 2011.

[49] Oriol Valentín. *Theory of Discontinuous Lambek Calculus*. PhD thesis, Universitat Autònoma de Barcelona, Barcelona, 2012.

[50] Oriol Valentín, Daniel Serret, and Glyn Morrill. A Count Invariant for Lambek Calculus with Additives and Bracket Modalities. In Glyn Morrill and Mark-Jan Nederhof, editors, *Proceedings of Formal Grammar 2012 and 2013*, volume 8036 of *Springer LNCS, FoLLI Publications in Logic, Language and Information*, pages 263–276, Berlin, 2013. Springer.

[51] J. van Benthem. *Language in Action: Categories, Lambdas, and Dynamic Logic*. Number 130 in Studies in Logic and the Foundations of Mathematics. North-Holland, Amsterdam, 1991. Revised student edition printed in 1995 by the MIT Press.

How WIE 'how' as Intensifier Co-occurs with other Intensifiers in German Sentence

Michael Richter and Roeland van Hout

Radboud University, Nijmegen

In this paper we state that in exclamative sentences in German, sentence inital *wie* 'how' which is an instance of degree related adverbial wh-exclamatives (Nouwen and Chernilovskaya [7]), is an intensifier of gradable adjectives (see Rett [8] on degrees within exclamatives). In (1) *wie* 'how' intensifies the gradable adjective *cool*:

(1) Wie cool ist das denn!
'how cool was that!'

Our second statement is that *wie* 'how' can co-occur with (gradable adjectival) intensifiers[1] of similar semantic properties, while it cannot with intensifiers with different properties which is exemplified in (2)a and 2(b), respectively. The words in italics in (2) are intensifiers of the gradable adjective cool. Note the *wie* does not intensify the bare adjective *cool* but an intensifier-adjective-complex such as *voll cool* 'totally cool':

(2) a. Wie *unglaublich/ übelst/voll* cool war das denn!!
'how incredibly/totally cool was that!'

b. *Wie *sehr/äußerst* cool war das denn!
'how very/extremely cool was that!'

Gutzmann and Turgay [3] distinguish between degree words in (2)b such as *sehr* 'very' and, in our interpretation, *äußerst* 'extremely' and expressive

[1](Morzycki [5]) states that sentences with (gradable) adjectival intensifiers are exclamatives as well. See (Nouwen [6]) on predicate status of adjectives in intensification.

intensifiers (=EI) in (2)a like *sau* 'totally, *voll* 'totally' and *übelst* 'totally' and argue that EI express a higher degree of intensifying than degree words.[2]

We start our analysis with the definition of adjectives. Following Seuren [9] adjectives are a binary relation between degrees and properties: x has a quality and this quality exceeds a degree d (see Kennedy and McNally [4]: Beck [1]). Carpenter [2] formalizes the combination of (gradable) adjectival predicates with intensifiers: Let P be a set of individuals, G a gradable (adjectival) predicate, then $Intensifier(G)(P)(y)$ holds iff $y \in P$ and there is a degree d such that $G(d)(y)$. That is to say, y has the property denoted by G to the degree d and only few elements $x \in P$, have the property denoted by G to the degree d, compared to the set of individuals P. The definition of intensifiers INT is given in (1):

(3) $INT \stackrel{def}{=} \lambda P.\lambda G.\lambda y P(y) \wedge some(\lambda d.G(d)(y) \wedge few(P)(\lambda x.G(d)(x)))$

The difference between types of intensifiers is represented by the predicate few which is a binary relation between the comparison set P and those $x \in P$ which have the quality denoted by G. The cardinality of the extension of the binary predicate few in (3) varies depending on the type of intensifiers. Let few_1 be the predicate of intensifiers such as *unglaublich*, few_2 the predicate of intensifiers such as *sehr* and few_3 the predicate of *wie*, and $x \neq y$ then (2) and (3) hold for a model M and a value assignment g (in (5) we use few as a shorthand notation for $|[[few(P)(\lambda x.G(d)(x))]]^{M,g}|$):

(4) $|[[few_{\mathrm{x}}(P)(\lambda x.G(d)(x))]]^{M,g}| \neq |[[few_{\mathrm{y}}(P)(\lambda x.G(d)(x))]]^{M,g}|$

(5) $\Delta x = |[[few_1]]| - |[[few_3]]| \wedge \Delta y = |[[few_2]]| - |[[few_3]]| \wedge \Delta x < \Delta y$

Six combinations are possible with few_1, few_2 and few_3, but just the combination of intensifiers with predicates few_1 (type *unglaublich* 'incredible') and few_3 (type *wie* 'how') yields grammaticality. We conclude that this is due to similar cardinalities of the denotations of few_1 and few_3. In contrast, the cardinalities of the denotations of few_3 and few_2 (type *sehr*

[2] An additional discriminating property between degree words and EI is the former can be iterated while the latter cannot (Gutzmann and Turgay [3]): *das war sehr, sehr/ äußerst, äußerst cool* 'that was very, very/extremely, extremely cool' vs. **das war unglaublich, unglaublich/sau, sau/voll,voll/ übelst,übelst cool* 'that was incredible, incredible/totally,totally cool'

'very') and in addition of few_1 and few_2[3] differ to the extent that the respective intensifiers cannot co-occur. The combinatorial restrictions indicate a graduality of intensifiers on a scale.

References

[1] Sigrid Beck. "Comparison constructions". In: *Semantics: An international handbook of natural language meaning*. Ed. by Klaus von Heusinger, Claudia Maienborn, and Paul Portner. Vol. 2. Berlin: De Gruyter, 2012, pp. 1341 –1390.

[2] Bob Carpenter. *Type-Logical Semantics*. Cambridge/Mass.: MIT Press, 1997.

[3] Daniel Gutzmann and Katharina Turgay. "Expressive intensifiers in German: syntax-semantics mismatches". In: *Empirical Issues in Syntax and Semantics*. Ed. by Christopher Piñion. 9. Palgrave McMillan, 2012, pp. 149 –166.

[4] Christopher Kennedy and Louise McNally. "Scale structure, degree modification, and the semantics of gradable predicates". In: *Language* 81 (2005), pp. 345 –381.

[5] Marcin Morzycki. "Adverbial modification of adjectives: Evaluatives and a little beyond". In: *Event Structures in Linguistic Form and Interpretation*. Ed. by Johannes Dölling and Tatjana Zybatow. Berlin and New York: Mouton de Gruyter, 2008.

[6] Rick Nouwen. "Degree modifiers and monotonicity". In: *Vagueness and Language Use*. Ed. by Paul Ègre and Nathan Klinedinst. Palgrave McMillan, 2011, pp. 146 –164.

[7] Rick Nouwen and Anna Chernilovskaya. "Two types of wh-exclamatives". In: *Linguistic Variation* 15.2 (2015), pp. 201 –224.

[8] Jessica Rett. "Exclamatives, degrees and speech acts". In: *Linguistics and Philosophy* 34.5 (2015), pp. 411 –442.

[9] Pieter A. M. Seuren. "The Comparative". In: *Generative Grammar in Europe*. Ed. by F. Kiefer and N. Ruwet. Dordrecht: Reidel, 1973, pp. 528 –564.

[3] Intensifiers with predicates few_1cannot combine with intensifiers with few_2, witness for instance **das ist sehr unglaublich cool* 'that is very incredible cool'

Language Processing Components of the StaViCTA Project

Maria Skeppstedt[†*], Kostiantyn Kucher[*], Carita Paradis[‡], Andreas Kerren[*]

The StaViCTA project is concerned with visualising the expression of stance in written text, and is therefore dependent on components for stance detection. These components are to (i) download and extract text from any HTML page and segment it into sentences, (ii) classify each sentence with respect to twelve different, notionally motivated, stance categories [3], and (iii) provide a RESTful HTTP API for communication with the visualisation components. The stance categories are CERTAINTY, UNCERTAINTY, CONTRAST, RECOMMENDATION, VOLITION, PREDICTION, AGREEMENT, DISAGREEMENT, TACT, RUDENESS, HYPOTHETICALITY, and SOURCE OF KNOWLEDGE.

Since standard libraries (jusText, Flask[1] and NLTK [1]) could be used for (i) and (iii), most work was spent on constructing machine learning classifiers (ii). These were trained on data that was created by manual text annotation of political blogs.

The twelve categories are not trivial to determine by human annotators (as shown by low inter-annotator agreement scores), and some of them occur rarely in most types of text [2]. This indicates that large resources in the form of annotated data would be required to train the classifiers, and for this reason active learning was applied [8]. The unlabelled sample closest to the separating hyperplane of a support vector machine was actively selected, i.e., an approach which had previously been shown to reduce the amount of training data required to detect similar categories [7]. The approach was implemented with the MongoDB[2] database and Scikit-learn's SVC class [5].

An annotation tool, developed within StaViCTA [4], was used to manually categorise the actively selected sentences with respect to the categories studied. In addition to this sentence-level annotation, the words that were used for expressing the categories were also marked. These were first automatically pre-annotated using the PAL tool [6] and then imported into

[†]Corresponding author, maria.skeppstedt@lnu.se
[*]Department of Computer Science, Linnaeus University, Växjö, Sweden
[‡]Centre for Languages and Literature, Lund University, Lund, Sweden
[1]`corpus.tools/wiki/Justext` and `flask.pocoo.org`
[2]`www.mongodb.com`

BRAT [9] and checked by an annotator. The word-level annotated data was then used for training a Scikit-learn LogisticRegression classifier to perform the stance-detection task (which in general led to better results than when using the SVC classifier). The probability scores of the logistic regression model could also be used to provide confidence estimates for the stance classification.[3]

References

[1] S. Bird. Nltk: The natural language toolkit. In *Proceedings of the Workshop on Effective Tools and Methodologies for Teaching NLP and CL*, Stroudsburg, PA, USA, 2002. ACL.

[2] Forthcoming. Annotating speaker stance in discourse: the Brexit Blog Corpus. Submitted for review, 2017.

[3] D. Glynn and M. Sjölin. *Subjectivity and epistemicity : corpus, discourse, and literary approaches to stance*. Centre for Languages and Literature, Lund University, Lund, 2014.

[4] K. Kucher, A. Kerren, C. Paradis, and M. Sahlgren. Visual Analysis of Text Annotations for Stance Classification with ALVA. In *Proceedings of EuroVis 2016 - Posters*, pages 49–51, Geneva, Switzerland, 2016. Eurographics.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[6] M. Skeppstedt, C. Paradis, and A. Kerren. PAL, a tool for Pre-annotation and Active Learning. *JLCL*, 31(1):91–110, 2016.

[7] M. Skeppstedt, M. Sahlgren, C. Paradis, and A. Kerren. Active learning for detection of stance components. In *Proceedings of PEOPLES*, pages 50–59, Stroudsburg, PA, USA, December 2016. ACL.

[8] M. Skeppstedt, V. Simaki, C. Paradis, and A. Kerren. Detection of stance and sentiment modifiers in political blogs. In *Proceedings of SPECOM (accepted)*, 2017.

[9] P. Stenetorp, S. Pyysalo, G. Topic, T. Ohta, S. Ananiadou, and J. Tsujii. BRAT: a web-based tool for NLP-assisted text annotation. In *Proceedings of EACL*, pages 102–107, Stroudsburg, PA, USA, 2012. ACL.

[3] StaViCTA is funded by the framework grant "the Digitized Society – Past, Present, and Future" with No. 2012-5659 from the Swedish Research Council (Vetenskapsrådet).

Vector Semantics for Lambek Calculus with Contraction

Gijs Wijnholds and Mehrnoosh Sadrzadeh
Queen Mary University of London
g.j.wijnholds@qmul.ac.uk, m.sadrzadeh@qmul.ac.uk

August 6, 2017

There are linguistic phenomena that, next to the movement of constituents, involve a specific form of copying and/or deletion of information, as argued for in [2, 6, 7]. Notable examples are pronoun relativisation [5, 6] and parasitic gaps [6], iterated coordination [7] and ellipsis [2]. Whether the burden of such phenomena should lie on the lexicon or on the syntactic process depends on the particularities of the phenomenon in question and the language in which it manifests itself. For instance, there are cases of pronoun relativisation that exemplify a form of derivational ambiguity ("mannen die vrouwen haten" in Dutch) whereas the English equivalent does not occur as an ambiguity but rather comes with a long distance dependency dealt with by different lexical instances (viz. compare "men who hate women" and "men whom women hate").

The case of (verb) ellipsis traditionally has been approached both as a syntactic problem as well as a semantic one [4]. Recent work in distributional semantics has shown that Frobenius algebras over vector spaces can deal with ellipsis [3] and relative pronouns [1]. However, they are introduced as meaning postulates which means that they can arise only through lexical semantics and not as part of the syntactic process.

Similar to the approaches of [2, 6, 7], we argue for the use of controlled forms of copying in a type logical system, so that we regain the balance between derivational and lexical ambiguity, in the presence of these complex semantic operations. In order to facilitate a controlled form of copying, we define an enrichment of the Lambek Calculus with control modalities that facilitates contraction on modally decorated formulas. The presence of such a structural rule in an uncontrolled/global version would obliterate the distinguishing power of the original calculus, whereas the use of modalities gives explicit control over when these operators are licensed.

We show how the Frobenius algebras used in previous work [1, 3] provide vector semantic counterparts for the proof theoretic copying rule of our system. Moreover, we show how our system relates to the systems of

[2, 6, 7]. The vector space semantics is given in the style of a Curry-Howard annotation: words are interpreted as vectors, proofs as linear maps between vector spaces. Semantic content is obtained from big data (not even limited to textual data), on top of which we describe the meaning of phrases beyond words by compositional interpretation from syntax into semantics. We illustrate with vector computations for ellipsis cases like "Mary writes Prolog and Gary does too" and "Gary loves his car and Bill too".

References

[1] Stephen Clark, Bob Coecke, and Mehrnoosh Sadrzadeh. The frobenius anatomy of relative pronouns. In *The 13th Meeting on the Mathematics of Language*, page 41, 2013.

[2] Gerhard Jäger. A multi-modal analysis of anaphora and ellipsis. *University of Pennsylvania Working Papers in Linguistics*, 5(2):2, 1998.

[3] Dimitri Kartsaklis, Matthew Purver, and Mehrnoosh Sadrzadeh. Verb phrase ellipsis using frobenius algebras in categorical compositional distributional semantics. *DSALT Workshop, European Summer School on Logic, Language and Information*, 2016.

[4] Ruth Kempson, Ronnie Cann, Eleni Gregoromichelaki Arasheshghi, and Matthew Purver. Ellipsis. *Chapter 4 of The Handbook of Contemporary Semantic Theory*, 3:114, 2015.

[5] Michael Moortgat. Multimodal linguistic inference. *Journal of Logic, Language and Information*, 5(3-4):349–385, 1996.

[6] Glyn Morrill and Oriol Valentín. Computational coverage of tlg: Nonlinearity. In *Proceedings of NLCS'15. Third Workshop on Natural Language and Computer Science*, volume 32, pages 51–63. EasyChair Publications, 2015.

[7] Glyn Morrill and Oriol Valentín. On the logic of expansion in natural language. In *Logical Aspects of Computational Linguistics. Celebrating 20 Years of LACL (1996–2016) 9th International Conference, LACL 2016, Nancy, France, December 5-7, 2016, Proceedings 9*, pages 228–246. Springer, 2016.

Variable Handling in DRT and DTS

Yukiko Yana[1,(✉)], Koji Mineshima[1,2], Daisuke Bekki[1,2]
[1]Ochanomizu University, Japan
yana.yukiko@is.ocha.ac.jp
[2]CREST, Japan Science and Technology Agency, Japan

Abstract

This paper discusses the differences in the behaviors of Discourse Representation Theory (DRT) and Dependent Type Semantics (DTS) with respect to variable handling. Since substitution in DRT is only partially defined, so is β-reduction and α-conversion, which brings about the *overwrite problem* (it is also called *destructive update*) and the *duplication problem*. We also compare the status of DRT with that of DTS and show that these two problems do not arise in DTS.

1 Introduction

Formal semantic frameworks designed to deal with discourse phenomena within a representationalist theory of interpretation include Discourse Representation Theory (DRT) [13, 14] and those based on Dependent Type Theory [19]. There are various proposals that extend and refine the original version of DRT, including Compositional DRT [20], an extension by Relational DRS [27, 26] and λ-DRT [4, 15]. Applications of Dependent Type Theory to natural language semantics originated from the pioneering work by G. Sundholm and A. Ranta [25, 23], followed by recent work on Modern Type Theory (MTT) [16], Type Theory with Records (TTR) [5], and Dependent Type Semantics (DTS) [2, 3].

It has been often stated that these two semantic frameworks, DRT and those based on Dependent Type Theory, are equivalent and mutually exchangeable [1, 9]. However, we will argue that these two kinds of frameworks differ in the process of deriving semantic representations and the behavior of the theory itself.

More specifically, as we will see later in detail, DRT and its compositional extensions do not provide a strict definition of α-conversion, so that substitution of terms remains partial and incomplete. Accordingly, there is a risk that two crucial notions in computational semantics that depend on substitution, namely, β-conversion and inference rules for quantification, will be only partially defined and incomplete.

The contribution of this paper is to provide a comparison between DRT (with extensions) and DTS, and to reveal differences in their semantic analyses and

derivation processes that result in theories that behave differently. As mentioned above, there are various semantic frameworks using dependent types, but it is rare to find work on a detailed mechanism of a compositional mapping from syntactic structures to semantic representations based on dependent types.[1] By contrast, DTS provides such a compositional semantics. This is why we will adopt DTS as a representative framework based on dependent types throughout this paper.

Although we will focus on a comparison between DRT and dependent types, there is an alternative to DRT that uses simply-typed λ-calculus for handling discourse dynamics (de Groote [7]). There are also various proposals on variable handling in a dynamic setting, specifically, those that can be subsumed as descendants of Dynamic Predicate Logic (DPL) (Groenendijk and Stokhof [10]). See, e.g., Vermeulen [28], Dekker [8], Groenendijk, Stokhof & Veltman [11], and, Nouwen [22]. A problem with the systems based on DPL is that they usually lack a proof-theoretic component. For this reason, we will confine our discussion to representational and inferential (proof-theoretic) systems in the tradition of DRT and dependent types.

The structure of this paper is as follows. In Section 2 we introduce the classical DRT and its various extensions that were proposed in the last 20 years. In Section 3 we expose two problems caused by the divergence between DRT's operations and those of ordinary logics, namely the *overwrite problem* and the *duplication problem*. In Section 4 we introduce the framework of DTS, and see how it can solve the overwrite problem and the duplication problem, which is an advantage of DTS over DRT. We close the paper with conclusions in Section 5.

2 Discourse Representation Theory

Since mid-1990's, *DRT* has been a name of a family of frameworks that extend the original, non-compositional theory proposed by Kamp [13] and Kamp and Reyle [14] in a compositional way. In this paper, we call the latter *Classical DRT* to distinguish it from the former.

2.1 Classical DRT

Classical DRT is known to be non-compositional theory in two senses: It is *intersententially* non-compositional because each sentence is not assigned a discourse representation structure (DRS), which can be determined only relative to its preceding discourse. It is also *intrasententially* non-compositional because each phrase is not assigned a DRS, whose contribution to the whole DRS is determined only relative to the surrounding syntactic tree and its preceding discourse.

[1] A notable exception is Dynamic Categorial Grammar (DCG) presented in Martin [17] and Martin and Pollard [18], where dependent type theory is adopted in combination with the distinction between phenogrammar and tectogrammar in a similar way to λ-Grammar (Muskens [21]) and ACG (de Groote [6]). We will leave comparison between DTS and DCG for another occasion.

Unlike Classical DRT, the extended frameworks such as Compositional DRT [20] (henceforth CDRT), Relational DRS [27], and λ-DRT [4] adopt a method of constructing DRS of an entire discourse from sentential DRSs by composing them by merge operations (thus intersententially compositional) and of a sentence from lexical DRSs by composing them in a bottom-up manner (thus intrasententially compositional). They share the basic idea in common, that is, combining Classical DRT with λ-operator and the operation of functional application, but are different in their way of implementing it. We will briefly review the definition of each representation system (CDRT, Relational DRS, and λ-DRT) and show how the compositional derivation of a DRS works in each system.

2.2 Compositional DRT

DRSs in CDRT are used as an abbreviation for following relation.

Definition 1 (DRS in Compositional DRT)**.** Let i, j be state variables.

u_1 ... u_n
Cond_1
...
Cond_m

$$= \lambda i \lambda j (i[u_1, ..., u_n]j \wedge Cond_1(j) \wedge ... \wedge Cond_m(j))$$

The merge operation of DRSs in CDRT is defined as Definition 2, and from this definition the following lemma is derived.

Definition 2. Let K_1, K_2 be DRS. $K_1; K_2 = \lambda i \lambda j \exists k (K_1(i)(k) \wedge K_2(k)(j))$

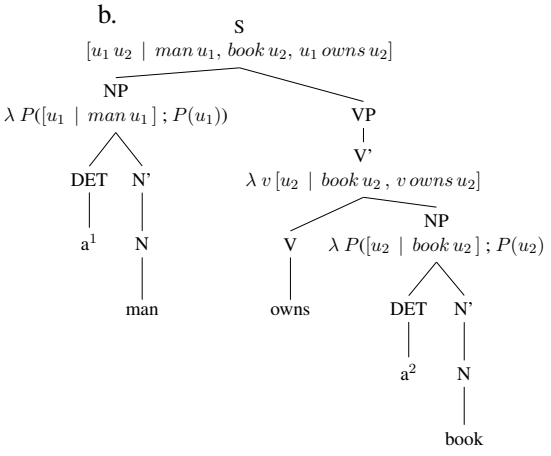
Lemma 3. Let $K_1 = (U_1, Cond_1), K_2 = (U_2, Cond_2)$ be DRS. If each discourse referent $u \in U_2$ does not occur in any of conditions $cond \in Cond_1$, then the merge of K_1 and K_2, written $K_1; K_2$, is defined as follows.

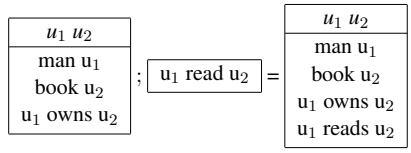
$$K_1; K_2 = [U_1 \cup U_2 \mid Cond_1 \cup Cond_2]$$

The example DRS construction is shown in (1).

(1) a. A^1 man owns a^2 book. He_1 reads it_2.

b.



c. 

Superscript and subscript numbers in the sentence (1a) respectively specify the numbers of discourse referents that they introduce and discourse referents of their antecedents. The trees (1b) and (1c) respectively show the intrasentential and intersentential compositions of DRSs in CDRT. The major difference of CDRT and Classical DRT is that discourse referents in CDRT are syntactically constant symbols. Therefore, operations on variables such as substitution and renaming are not defined for discourse referents of CDRT.

2.3 Relational DRS

The definition of Relational DRS is slightly different from DRS of Classical DRT, as shown in Definition 4.

Definition 4 (DRS in Relational DRS)**.**

1. If v is a discourse referent, then v is a DRS.
2. If $t_1, \ldots, t_n$ are terms and P is an n-place predicate letter, then $P(t_1, \ldots, t_n)$ is a DRS.
3. If v is a discourse referent and t is a term, then $v = t$ is a DRS.
4. If D is a DRS, then $\neg D$ is a DRS.
5. Nothing else is a DRS.

The major difference of Relational DRS from Classical DRT is that discourse referents and predicates themselves qualify as DRSs according to Definition 4.1. and Definition 4.2. Based on this, the definition of Relational DRS (RDRS) is shown in Definition 5.

Definition 5 (RDRS)**.**

1. DRS D is RDRS.
2. If R is RDRS, then $\neg R$ is RDRS.
3. If R_1, R_2 are RDRS, then $R_1 \bullet R_2$ is RDRS.
4. Nothing else is a RDRS.

Definition 5 shows that complex RDRSs are obtained by joining DRS with join operator "$\bullet$". The reduction operation (in the sense of β-reduction) for complex RDRSs are defined by the set of reduction rules, a part of which is exemplified in (2) :

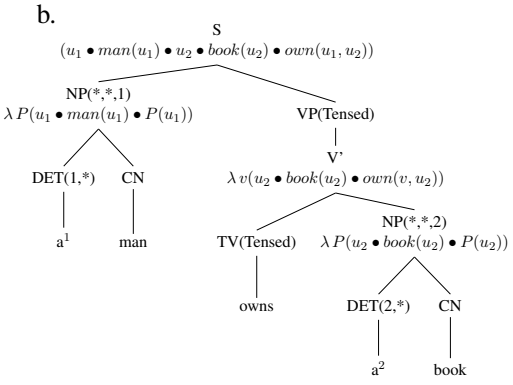
(2) Let R, R' be RDRS, v, w be discourse referents:

$$((R \bullet v) \bullet R') \Rightarrow ((R; v); R')\ (v \notin R)$$
$$((R; w); [w/v]R')(v \in R, w \notin R')$$

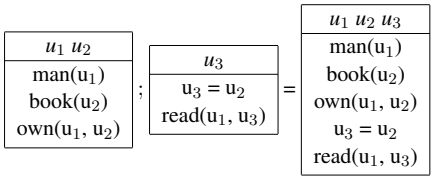
where the operator ";" is defined in the same way as Definition 2. The notation $[y/x]R$ is the result of substituting y to x in R. By reduction, the discourse (3a) is converted to the DRS in (4).

(3) a. A^1 man owns a^2 book. He_1 reads it_2^3.

b.



(4)



Consequently, the extension by RDRS also achieves the intrasentential and intersentential compositionality.

2.4 λ-DRT

Among several versions of λ-DRT, we discuss the one formulated by Kuschert et al. [15], which introduces δ-operator, the binder of discourse referents. The definition of the merging operations of DRSs is defined as Definition 6, where $\otimes$ and ; are intrasentential and intersentential merge operators, respectively.

Definition 6 (Merge in λ-DRT)**.**

1. $\delta X.\mathrm{C}_1 \otimes \delta Y.\mathrm{C}_2 \rightarrow \delta(X \cup Y).\ \mathrm{C}_1 \wedge \mathrm{C}_2$
2. $\delta X.\mathrm{C}_1\ ;\ \delta Y.\mathrm{C}_2 \rightarrow \delta(X \cup Y).\ \mathrm{C}_1 \wedge \mathrm{C}_2$

An example of DRS construction using this definition is shown as (5).

(5) a. A man owns a book. He read it.

b.

$(\text{own}^a@(\text{a}^j@\text{book}))@(\text{a}^i@\text{man})$

$= (\lambda Q.\lambda v.(Q(\lambda u.\delta\{e_a\}.own(e_a,u,v)))$
$@(\lambda P.\lambda Q.(\delta\{x_j\}.T \times P(x_j) \times Q(x_j))$
$@\lambda u.\delta\{\}.book(u)))$
$@(\lambda P.\lambda Q.(\delta\{x_i\}.T \times P(x_i) \times Q(x_i))$
$@\lambda u.\delta\{\}.man(u))$

$\rightarrow_\beta$

$e_a\ x_i\ x_j$
man(x_i)
book(x_j)
own(e_a, x_i, x_j)

c.

$e_a\ x_i\ x_j$
man(x_i)
book(x_j)
own(e_a, x_i, x_j)

; | read(e_b, x_i, x_j) |

$\rightarrow_\delta$

$e_a\ x_i\ x_j$
man(x_i)
book(x_j)
own(e_a, x_i, x_j)
read(e_b, x_i, x_j)

Note that each discourse referent in (5c) has a unique name. In λ-DRT, only those with unique names are treated as well-formed formulae (wff) as defined as Definition 7.

Definition 7 (wff in λ-DRT)**.** Let A be DRS. $A \in$ wff iff for every A's sub-expression of the forms $A_1 \otimes A_2$, $A_1; A_2$, $A_1(A_2)$, A_1 and A_2 do not share the same discourse referents.

According to Definition 7, each discourse referent has a unique name, which makes safe the union operation in (6).

3 Two problems about variable handling in DRT

In this section, we will see that DRT operations such as substitution, α-conversion, and β-reduction behave differently from those in standard type theories. This will bring about two problems: the overwrite problem and duplication problem. Although these problems have been recognized and discussed in the literature, we will mainly focus on the issues of variable handling and provide a survey of problematic cases of a family of DRT frameworks from that perspective.

In Section 3.1, we discuss the overwrite problem and see how it arises in each theory we reviewed in Section 2. In Section 3.2, we discuss the duplication problem and consider why it arises through examining the relationship between substitution and binding scope. In Section 3.3, we analyze the logical structures of DRT that cause these two problems.

3.1 Overwrite problem

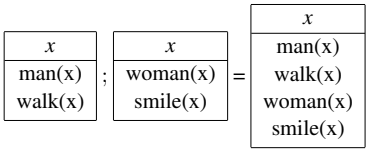
The overwrite problem of DRT, first pointed out by Zeevat [30], is that there is a case where a link between a discourse referent and an anaphoric expression unintentionally gets destroyed as a result of the merge operation. In Zeevat [30], the merge operation is defined as Definition 8.

Definition 8. Let $K_1 = (U_1, Cond_1)$ and $K_2 = (U_2, Cond_2)$ be a DRS.

$$merge(K_1, K_2) = (U_1 \cup U_2, Cond_1 \cup Cond_2)$$

A problematic case where the overwrite problem occurs is exemplified by the example (6).

(6) a. A man walked. A woman smiled.

b. 

In (6a), *a man* and *a woman* denote different persons, but they are interpreted as the same person in DRS (6b) as a result of merging. Zeevat [30] pointed out that discourse referents to be introduced should be restricted to avoid such problematic cases.

In the extended DRT frameworks in the previous section, this problem does not appear to happen since discourse referents to be introduced are assumed to be distinct with each other. However, this assumption does not save conjunction reduction cases such as the sentence (7) where a discourse referent for an indefinite NP gets copied.

(7) Bill and Sue own a donkey. (cf. Muskens [20])

(8) a. The donkey which Bill owns eats an apple.
b. The donkey which Sue owns eats an apple.

The sentence (7) is equivalent to the conjunction that Bill owns a donkey and Sue owns a donkey, thus there may be two different donkeys. Now, consider two discourses where the sentence (7) is followed by the sentence (8a) and the sentence (8b) and how each extended DRT fails to give an appropriate DRS to those mini-discourses.[2]

3.1.1 The case of CDRT

The DRS for the mini-discourse (9) in terms of CDRT is given as (10).

(9) Bill[1] and Sue[2] own a[3] donkey.
The$_3$ donkey which Bill[1] owns eats an[4] apple.

[2] It might be objected that the sentence (7) should not be analyzed as an instance of conjunction reduction. However, conjunctive constructions like (i), where a sentential adverbial such as *possibly* and *probably* applies to a conjunct NP (cf. Zamparelli [29]), would pose a challenge to such a view.

(i) Bill and possibly Sue own a donkey.

(10)

u_3
donkey u_3
Bill owns u_3

;

u_3
donkey u_3
Sue owns u_3

;

u_4
donkey u_3
Bill owns u_3
apple u_4
u_3 eats u_4

According to the Definition 2, the leftmost and the center DRSs of (10) are merged first. This two DRSs merging results in (11).

(11) $\lambda i \lambda j. \exists k(i[u_3]k \wedge (donkey\ u_3)(k) \wedge (Bill\ owns\ u_3)(k)$
$\wedge k[u_3]j \wedge (donkey\ u_3)(j) \wedge (Sue\ owns\ u_3)(j))$

In (11), discourse referent u_3 refers to Bill's donkeys and Sue's donkeys. However, these donkeys cannot be distinguished because they are represented by the same discourse referent.

Syntactically, discourse referents in CDRT are constant symbols, so there is no way to change the names of discourse referents. As a result, it was impossible to rename the discourse referent u_3 or distinguish the two occurrences of u_3.

This problem was first pointed out by Haug [12], but no solution is ever proposed to this problem so far.

3.1.2 The case of Relational DRS

The DRS for the mini-discourse (12) in terms of Relational DRS is given as (13).

(12) Bill_1 and Sue_2 own a^3 donkey.
The^4_3 donkey which Sue_2 owns eats an^5 apple.

(13)

u_3
u_1 = b
donkey(u_3)
own(u_1, u_3)

•

u_3
u_2 = b
donkey(u_3)
own(u_1, u_3)

•

u_4
u_3 = u_4
donkey(u_4)
own(u_2, u_4)
apple(u_5)
eat(u_4, u_5)

According to Definition 2, we first merge the leftmost and the center DRSs of (13), then the merged DRS and the rightmost DRS are merged. When merging, the discourse referent that conflicts with others is substituted for a new discourse referent. Therefore, the result of the combination is as shown in (14b).

(14) a. (

u_3
u_1 = b
donkey(u_3)
own(u_1, u_3)

; [u_6/u_3]

u_3
u_2 = b
donkey(u_3)
own(u_1, u_3)

) ;

u_4
u_3 = u_4
donkey(u_4)
own(u_2, u_4)
apple(u_5)
eat(u_4, u_5)

b.

u_3 u_4 u_5 u_6
u_1 = b
donkey(u_3)
own(u_1, u_3)
u_2 = b
donkey(u_6)
own(u_1, u_6)
u_3 = u_4
donkey(u_4)
own(u_2, u_4)
apple(u_5)
eat(u_4, u_5)

In (14b), u_4 and u_5 denote Bill's donkey and an apple respectively, so "eat(u_4, u_5)" means that Bill's donkey eats an apple. However, this does not capture the truth-condition of (12) where Sue's donkey eats an apple.

Since the substitution of RDRS considers only the RDRSs that immediately precedes or succeed it, the scope of substitution in (14a) only contains the central DRS. However, considering the meaning of the sentence (12), u_3 in the rightmost DRS must be substituted as well, so substitution defined in this way does not work well.[3]

Note that widening the scope of substitution is not a good strategy: if u_3 in the rightmost DRS is the same as u_3 introduced by the leftmost DRS, it leads to another incorrect derivation.

More generally, this result can be regarded as an instance of the following phenomenon (15):

(15) $R_1 \bullet \cdots \bullet R_k \bullet \cdots \bullet R_n$
$\neq R_1 \bullet \cdots \bullet [u_i/u_j]R_k \bullet \cdots \bullet R_n$

In most logics and type theories, two logical expressions whose only difference is a variable name is α-equivalent, so the contents of the expression itself do not change even if we replace the subexpression like (15). Under this extension by RDRS, however, the scope of binding by discourse referent is taken wider than ordinary logic, so the contents of the expression will be different if we do such a replacement. Therefore, we found that this extension leads to a theory in which renaming and substitution themselves are defined but α-conversion cannot be performed.

3.1.3 The case of λ-DRT

The DRS for the mini-discourse (16) in terms of λ-DRT is given as (17).

(16) Bill^i and Sue^j own^a a^k donkey.
The^l_k donkey which Sue^m owns^b eats^c an^n apple.

(17)

$x_i\ x_k\ e_a$
x_i = j
donkey(x_k)
own(e_a, x_i, x_k)

;

$x_j\ x_k\ e_a$
x_j = s
donkey(x_k)
own(e_a, x_j, x_k)

;

$x_l\ x_m\ x_n\ e_b\ e_c$
$\mathrm{x}_l = \mathrm{x}_k$
donkey(x_l)
x_m = s
own(e_b, x_m, x_l)
apple(x_n)
eat(e_c, x_l, x_n)

Since the discourse referents x_k and e_a in the leftmost and the center DRS in (17) conflict, the DRS in (17) is not well-formed according to Definition 7. Note

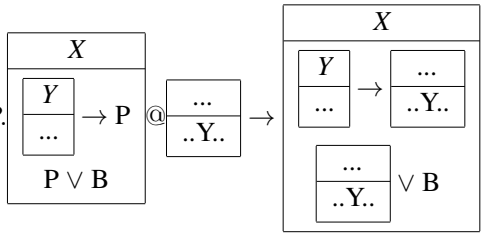
[3] van Eijck [26] proposed a framework to solve this problem by introducing the *de Bruijn*-style notation to DRT. However, the theory does not provide lexical items, and it is not clear how to extend this theory into *intrasententially* compositional settings. It is one of our future work to investigate a possibility of lexicalizing van Eijk's theory and compare it with ours, but this is beyond the scope of our paper.

that examples like (17) are not artificially created but derived from natural language examples such as (16). This means that the theory undergenerates for a conjunction reduction case likes (16), which is a problem for λ-DRT.

The version of λ-DRT presented in Kuschert et al. [15] attempts to restrict the range of its application by introducing the notion of "sensible expression" and "substitutability", and so on. It is claimed that such a theory with additional constraints does not cause a problem in practice Kuschert et al. [15]; however, the above discussion shows that there actually exists a problematic example in natural language.

3.2 Duplication problem

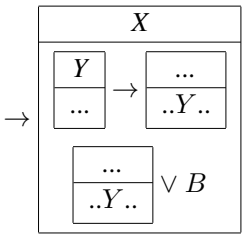
The duplication problem pointed out in Kuschert et al. [15] is that the binding status of variables in DRS can change after β-reduction during the process of building the representation of a sentence.

(18) $\lambda P.$ 

There are two occurrences of Y in the reduced DRT in (18), one is bound and another is free. This structure is ill-formed since the binding status of the different occurrences of the same discourse referent Y differ. However, Kuschert et al. [15] mentioned that this derivation cannot be avoided by judging it from the surface structure of the sentence.

Let us examine whether the duplication problem is avoided in other extended DRTs. CDRT yields the same result as λ-DRT does since their behavior is the same with respect to β-reduction. In RDRS, this intrasentential merging takes place at the level of RDRS, which is exemplified in (19) where $R \rightarrow R'$ is defined as $\neg(R \bullet \neg R')$.

(19) $\lambda P.(X \bullet \neg(Y \bullet ... \bullet \neg P) \bullet P \vee B)(..Y..) \rightarrow (X \bullet \neg(Y \bullet ... \bullet \neg(..Y..)) \bullet (..Y..) \vee B)$



As shown in (19), the result ends up in the same DRS as λ-DRT and CDRT, despite the differences in the derivation processes. Therefore, all the extended DRTs discussed in this paper fall into the duplication problem.

The reason for allowing such derivation is that there is no constraint on β-reduction: It is a standard assumption in λ-calculus that one has to check free variables during β-reduction. However, such a constraint would bring about another problem in constructing DRS, because of the non-standard notion of binding in the extended DRT such as λ-DRT called *dynamically bound* variables. A variable x is *dynamically bound* iff x is a free variable within an argument given to a function in which the variable corresponding to its argument appears within the scope of x introduced by the universe of some DRS. This is exemplified by the example (21), the DRS of (20), where x in walk(x) is dynamically bound.

(20) A man walks.

(21) $\lambda P.$ $\begin{array}{|c|}\hline x \\ \hline \text{man(x)} \\ P \\ \hline \end{array}$ @ $\boxed{\text{walk(x)}}$ $\rightarrow$ $\begin{array}{|c|}\hline x \\ \hline \text{walk(x)} \\ \hline \end{array}$

The variable x of walk(x) is free, but it is dynamically bound and not a free variable after β-reduction. Such bindings are common in extended DRTs.

In most λ-calculi, when variables in the argument conflict the variables in a functional expression, they force renaming of the variables in the function to fresh ones. In the case of extended DRTs, however, such renaming is impossible, since (20) wound become (22) when introducing such a renaming constraint.

(22) A man walks.

(23) $\lambda P.[y/x]$ $\begin{array}{|c|}\hline x \\ \hline \text{man(x)} \\ P \\ \hline \end{array}$ @ $\boxed{\text{walk(x)}}$ $\rightarrow$ $\begin{array}{|c|}\hline y \\ \hline \text{man(y)} \\ \text{walk(x)} \\ \hline \end{array}$

In (23), man's argument y and walk's argument x are different, thus this derivation is incorrect. In other words, the notion of variable renaming during β-reduction, which is widely assumed in λ-calculus, and the notion of dynamically bound variables are not compatible with each other. Therefore, there does not seem to be a straightforward remedy to the duplication problem given the theoretical setting of the extended DRTs which we discussed in this paper.

3.3 Logical structure of DRT

Both of the problems pointed out in this section are caused by the logical structure of DRT: In standard logic, since the scope of substitution agrees with that of binding, all variables under the scope are substituted. However, because the scope of substitution is wider than binding scope in DRT, α-conversion and substitution in standard logic style would destroy the binding relations. Therefore, it is necessary

to restrict the definition of substitution. Since the discourse referents in CDRT are constant symbols, it does not define substitution for them. Relational DRS defines substitution in consideration of free variables in immediately following RDRS. λ-DRT defines substitutability for the definition of substitution.

Due to these restrictions on substitution, β-reduction is not fully defined in the extended DRTs. The duplication problem shown in 3.2 is an instance of this problem. Since β-reduction plays an important role in composing DRS, it is a major problem that the safety of β-reduction is not guaranteed. If compositionalitiy depends on β-reduction, it should be reconsidered whether these extended DRTs are compositional theories.

Also, as a result of restricting substitution in order to keep binding relations, the binding relations of anaphora cannot be derived correctly. The overwrite problem shown in 3.1 is an instance of this problem. The anaphora resolution is one of the important task in formal semantics, but there are cases that are not solved by the extended DRTs.

Why does this happen even though the extended DRTs are based on λ-calculus? The reason lies in the difference between discourse referents in DRT and variables in λ-calculus.

The discourse referents in DRT play a role of binding variables as variables in first-order logic do. For example, a discourse referent x in (24a) plays almost the same role as x of $\exists x$ does in (24b).

(24) a.

x
man(x)
walk(x)

b. $\exists x(man(x) \wedge walk(x))$

When the extended DRT tried to introduce the intrasentential compositionality into Classical DRT, they integrate λ-calculus into DRT, but then, while variables bound by λ-operators behave like the variables in first-order logic, discourse referents in extended DRTs behave differently, although they are bound by the same λ-operator, thus some property of λ-calculus is lost.

The discrepancy between the status of variables and discourse referents causes the failure of β-reduction despite using λ-calculus. To solve this problem, it is necessary for the discourse referents to share the same property as the binding variables in logic, but it is impossible as it will deprive them of their dynamic nature.

λ-DRT introduces a new operator δ to bind discourse referents, so it seems that such a discrepancy does not occur at first glance. However, in the first place, since λ-calculus does not have the operator δ, the property of λ-calculus differs from the original when they introduce such an operator.

4 Dependent Type Semantics

DTS (Bekki [2]; Bekki and Mineshima [3]) is a proof-theoretic semantics based on Dependent Type Theory (Martin-Löf [19]). As we argued in the previous section, there are difficulties with DRT in that operations in the ordinary logic such as substitution are not defined in an adequate way. In DTS, by contrast, α-conversion and β-conversion are defined in the standard type-theoretic manner without stipulating any additional constraints, so that the overwrite problem and the duplication problem do not arise in this framework. We will see that anaphora can be treated appropriately in the compositional framework of DTS.

4.1 Dependent Type Theory

Dependent type theory is an extension of the simple typed λ-calculus. It can be characterized by being able to treat types depending on terms. There are two important types in dependent type theory, Π-type and Σ-type. Π-type generalizes function type in simple typed λ-calculus. We write $(x : A) \to B(x)$ for a Π-type. Σ-type generalizes product type in simple typed λ-calculus. We write $(x : A) \times B(x)$ or $\left[\begin{array}{l} x : A \\ B \end{array} \right]$ for a Σ-type. The notation $x : A$ represents that the term x has a type A, and $B(x)$ represents a type dependent on the term x. Π-types and Σ-types correspond to universal quantification and existential quantification in logic. Due to this correspondence, the semantic representation of a sentence in natural language can be specified in terms of a *type*. A term having such a type that corresponds to a proposition is called a *proof term*.

Π-types and Σ-types have their own inference rules, in particular, formation rules, introduction rules and elimination rules. For instance, the following are elimination rules of Σ-type, where $\pi_i \, (i = 1, 2)$ is a projection function.

(25) a. $$\frac{M : \left[\begin{array}{l} x : A \\ B \end{array} \right]}{\pi_1 M : A} \; (\Sigma \text{ E})$$

b. $$\frac{M : \left[\begin{array}{l} x : A \\ B \end{array} \right]}{\pi_2 M : B[\pi_1 M / x]} \; (\Sigma \text{ E})$$

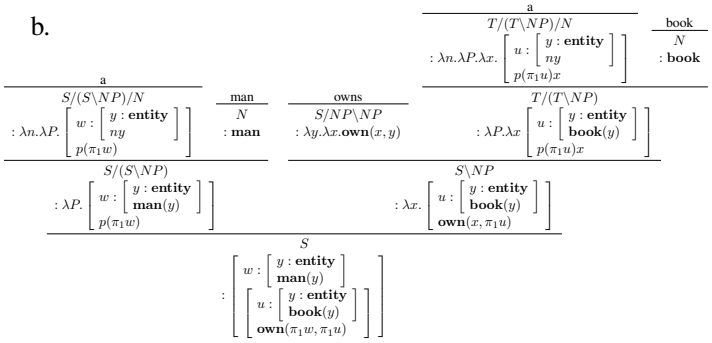
See e.g., Martin-Löf [19] for inference rules in Dependent Type Theory.

4.2 Anaphora resolution in DTS

Since dependent type theory has types that depend on terms, it can represent the meaning of a proposition that depends on the meaning of its previous context. The conjunction of semantic representations is defined in terms of Σ-types as in Definition 9.

(26) a. A man owns a book. He reads it.

b.



c.

$$\left[\begin{array}{l} w : \left[\begin{array}{l} y : \mathbf{entity} \\ \mathbf{man}(y) \end{array}\right] \\ \left[\begin{array}{l} u : \left[\begin{array}{l} y : \mathbf{entity} \\ \mathbf{book}(y) \end{array}\right] \\ \mathbf{own}(\pi_1 w, \pi_1 u) \end{array}\right] \end{array}\right] ; \mathbf{read}\left(\pi_1\left(@_1 : \left[\begin{array}{l} x : \mathbf{entity} \\ \mathbf{male}(x) \end{array}\right]\right), \pi_1\left(@_2 : \left[\begin{array}{l} x : \mathbf{entity} \\ \neg\mathbf{human}(x) \end{array}\right]\right)\right)$$

$$= \left[\begin{array}{l} v : \left[\begin{array}{l} w : \left[\begin{array}{l} y : \mathbf{entity} \\ \mathbf{man}(y) \end{array}\right] \\ \left[\begin{array}{l} u : \left[\begin{array}{l} y : \mathbf{entity} \\ \mathbf{book}(y) \end{array}\right] \\ \mathbf{own}(\pi_1 w, \pi_1 u) \end{array}\right] \end{array}\right] \\ \mathbf{read}\left(\pi_1 @_1 : \left[\begin{array}{l} x : \mathbf{entity} \\ \mathbf{male}(x) \end{array}\right], \pi_1 @_2 : \left[\begin{array}{l} x : \mathbf{entity} \\ \neg\mathbf{human}(x) \end{array}\right]\right) \end{array}\right]$$

Definition 9. Let M, N be semantic representations in DTS.

$$M; N = \left[\begin{array}{l} u : M \\ N \end{array}\right] \text{ (where } u \notin fv(N) \cup bv(N))$$

This definition provides a way to analyze intrasentential and intersentential anaphora in a unified compositional way.

As an illustration, consider the mini-discourse in (26a). A compositional derivation of the semantic representation for the first sentence is given in (26b), where we assume CCG [24] as a syntactic framework. Pronouns and other context-dependent expressions are represented by using *underspecified term* $@_i$. An underspecified term has an annotated type of the form $@_i A$. In (26a), for example, the pronoun he is represented as (27a) and the pronoun it as (27b).

(27) a. $\pi_1\left(@_1 : \left[\begin{array}{l} x : \mathbf{entity} \\ \mathbf{male}(x) \end{array}\right]\right)$

b. $\pi_1\left(@_2 : \left[\begin{array}{l} x : \mathbf{entity} \\ \neg\mathbf{human}(x) \end{array}\right]\right)$

Here the index in $@_1$ and $@_2$ is used to distinguish underspecified terms $@$, so its role is different from from that played in DRT.

The semantic representations of the first and second sentences are combined as shown in (26c), which gives the semantic representation for the whole discourse (26a). In DTS, the process of resolving anaphora is formalized as a process of type checking (cf. Bekki and Mineshima [3]). In the case of (26c), anaphora can be resolved by substituting the unspecified terms $@_1$ and $@_2$ with a term having the annotated type. Suppose that the background knowledge provides the axioms

$$f : (x{:}\,\mathbf{entity}) \to (\mathbf{man}(x) \to \mathbf{male}(x))$$

and

$$g : (x{:}\,\mathbf{entity}) \to (\mathbf{book}(x) \to \neg\mathbf{human}(x)).$$

Then a term having the annotated type $\left[\begin{array}{l} x : \mathbf{entity} \\ \mathbf{male}(x) \end{array}\right]$ for $@_1$ can be constructed as $(\pi_1\pi_1 v, f(\pi_1\pi_1 v)(\pi_2\pi_1 v))$. Similarly, a term having the annotated type $\left[\begin{array}{l} x : \mathbf{entity} \\ \neg\mathbf{human}(x) \end{array}\right]$ for $@_2$ can be constructed as $(\pi_1\pi_1\pi_2 v, g(\pi_1\pi_1\pi_2 v)(\pi_2\pi_2\pi_1 v))$. In this way, in DTS the process of anaphora resolution may involve an inference using background knowledge.

(28) Bill and Sue own a donkey.
The$_1$ donkey which Bill owns eats an apple.

(29)
$$\left[\begin{array}{l} p : \left[\begin{array}{l} u : \left[\begin{array}{l} x : \mathbf{entity} \\ \mathbf{donkey}(x) \end{array}\right] \\ \mathbf{own}(b, \pi_1 u) \end{array}\right] \\ \left[\begin{array}{l} v : \left[\begin{array}{l} y : \mathbf{entity} \\ \mathbf{donkey}(y) \end{array}\right] \\ \mathbf{own}(s, \pi_1 v) \end{array}\right] \end{array}\right] ; \left[\begin{array}{l} w : \left[\begin{array}{l} x : \mathbf{entity} \\ \mathbf{apple}(x) \end{array}\right] \\ \mathbf{eat}(\pi_1(@\left[\begin{array}{l} x : \mathbf{entity} \\ \left[\begin{array}{l} \mathbf{donkey}(x) \\ \mathbf{own}(b, x) \end{array}\right] \end{array}\right]), \pi_1 w) \end{array}\right] =$$

$$\left[\begin{array}{l} q : \left[\begin{array}{l} p : \left[\begin{array}{l} u : \left[\begin{array}{l} x : \mathbf{entity} \\ \mathbf{donkey}(x) \end{array}\right] \\ \mathbf{own}(b, \pi_1 u) \end{array}\right] \\ \left[\begin{array}{l} v : \left[\begin{array}{l} y : \mathbf{entity} \\ \mathbf{donkey}(y) \end{array}\right] \\ \mathbf{own}(s, \pi_1 v) \end{array}\right] \end{array}\right] \\ \left[\begin{array}{l} w : \left[\begin{array}{l} x : \mathbf{entity} \\ \mathbf{apple}(x) \end{array}\right] \\ \mathbf{eat}(\pi_1(@ : \left[\begin{array}{l} x : \mathbf{entity} \\ \left[\begin{array}{l} \mathbf{donkey}(x) \\ \mathbf{own}(b, x) \end{array}\right] \end{array}\right]), \pi_1 w) \end{array}\right] \end{array}\right]$$

By replacing these terms with $@_1$ and $@_2$ in the final semantic representation in (26c) and by computing the projection function π_1 with the rule $\pi_1(m, n) = m$, we can obtain the following semantic representation.

(30)
$$\left[\begin{array}{l} v : \left[\begin{array}{l} w : \left[\begin{array}{l} y : \mathbf{entity} \\ \mathbf{man}(y) \end{array}\right] \\ \left[\begin{array}{l} u : \left[\begin{array}{l} y : \mathbf{entity} \\ \mathbf{book}(y) \end{array}\right] \\ \mathbf{own}(\pi_1 w, \pi_1 u) \end{array}\right] \end{array}\right] \\ \mathbf{read}(\pi_1\pi_1 v, \pi_1\pi_1\pi_2 v) \end{array}\right]$$

This representation captures the intended meaning of the discourse in (26a).

DTS resolves anaphora by type checking, and DTS distinguish terms by the proofs, in other words, paths to the antecedents using operators π_1 and π_2, instead of their names. In the next subsection, we show how DTS avoids the overwrite problem and the duplication problem.

4.3 Overwrite problem

Let us consider the problematic case of a conjunction reduction construction that causes the overwrite problem discussed in Section 3. For instance, the semantic representations of the two sentences in (28) are combined as shown in (29). The type annotated with @ in (29) requires as its term a triplet (x, t, u) of a term x having type $\mathbf{entity}$, a proof term t of type $\mathbf{donkey}(x)$, that is, a proof term for the proposition that x is a donkey, and a proof term u of type $\mathbf{own}(b, x)$, that is, a proof term for the proposition that Bill owns x.[4] By type checking and proof construction, we may find the term $(\pi_1\pi_1\pi_1 q, (\pi_2\pi_1\pi_1 q, \pi_2\pi_1 q))$ having this type. By substituting the term @ with this term, we can obtain the following semantic representation.

(31)
$$\left[\begin{array}{l} q : \left[\begin{array}{l} p : \left[\begin{array}{l} u : \left[\begin{array}{l} x : \mathbf{entity} \\ \mathbf{donkey}(x) \end{array}\right] \\ \mathbf{own}(b, \pi_1 u) \end{array}\right] \\ \left[\begin{array}{l} v : \left[\begin{array}{l} y : \mathbf{entity} \\ \mathbf{donkey}(y) \end{array}\right] \\ \mathbf{own}(s, \pi_1 v) \end{array}\right] \end{array}\right] \\ \left[\begin{array}{l} w : \left[\begin{array}{l} x : \mathbf{entity} \\ \mathbf{apple}(x) \end{array}\right] \\ \mathbf{eat}(\pi_1\pi_1\pi_1 q, \pi_1 w) \end{array}\right] \end{array}\right]$$

In this way, a reading in which Bill's donkey substitutes the term @ is derived.

When the term @ refers to Sue's donkey, the term to be substituted must be a triplet (x, t, u') where x and t are the same as before and u' is a proof term of type $\mathbf{own}(s, x)$, that is, a proof term for the proposition that Sue owns x. We can construct a term $(\pi_1\pi_1\pi_2 q, (\pi_2\pi_1\pi_2 q, \pi_2\pi_2 q))$ satisfying this condition. As mentioned above, the term corresponding to Sue's donkey and the term corresponding to Bill's donkey are distinguished by their derivation path, that is, the way a proof term is

[4] To be precise, it is a pair of the term and a pair of proof terms for an unary predicate and a binary predicate, but it can be rearranged by elimination rule and introduction rule of Σ-type. So it can be treated as a triplet.

constructed, not by the name of the term. This way of distinguishing anaphoric dependencies makes possible to derive two ways of picking up an object introduced by the first sentence in (28).

4.4 Duplication problem

Recall that DRT suffers from the duplication problem as depicted in (18). (32) is the semantic representation of (18) in DTS. For the sake of exposition, we gives y subscripts 1 and 2 to distinguish its different occurrences

$$\text{(32)} \quad \lambda P. \left[\begin{array}{l} x : A \\ \left[\begin{array}{l}(y_1 : B) \rightarrow P\end{array}\right] \\ P \vee C \end{array}\right] (\left[\begin{array}{l} u : D \\ E(y_2) \end{array}\right]) \\ \rightarrow_\beta \left[\begin{array}{l} x : A \\ \left[\begin{array}{l}(y_1 : B) \rightarrow P\end{array}\right] \\ P \vee C \end{array}\right] [\left[\begin{array}{l} u : D \\ E(y_2) \end{array}\right] / P]$$

In (32), y_1 is a bound variable and y_2 is a free variable. Following simply-typed λ-calculus, DTS renames y_1 to avoid the variable-name crash, and obtains the semantic representation (33), where a fresh variable w is introduced.

$$\text{(33)} \quad \left[\begin{array}{l} x : A \\ \left[\begin{array}{l} (w : B) \rightarrow \left[\begin{array}{l} u : D \\ E(y) \end{array}\right] \\ \left[\begin{array}{l} u : D \\ E(y) \end{array}\right] \end{array}\right] \vee C \end{array}\right]$$

In (33), the duplication problem is well avoided because the binding status of the two occurrences of y is the same. DTS binds variables in the same way as simply-typed λ-calculus does, which contrast with the status of *dynamically bound* variables in DRTs. Therefore, renaming does not conflict with the linguistic analysis and it can define the binding relations in a clear way.

5 Conclusion

We showed that some features in the logical structure of DRT cause problems in the case of the extended, compositionalized DRT, and that they cannot avoid the problems as long as they adopt the approach mentioned in the previous sections. We also showed that these problems never happen in DTS.

Since DTS is based on dependent type theory, which is a natural extension of simply-typed λ-calculus, it keeps the property of binder-variable relation in λ-calculus intact. DTS also derives a semantic representation which uses typed variables rather than discourse referents. Therefore, DTS keeps the safety of β-reduction and α-conversion, which makes its logical structure robust.

To summarize, we have discussed that DRT and DTS have quite a different property with respect to their compositionality. It is stated that they are the same in the sense that the semantic representations of one analysis can be translated into

those of another, we should also be aware of their differences: there exist semantic representations of DTS whose corresponding DRSs are not compositionally derivable in DRT.

References

[1] René Ahn and Hans-Peter Kolb. Discourse representation meets constructive mathematics. In *Papers from the Second Symposium on Logic and Language*. Akademiai Kiado, 1990.

[2] Daisuke Bekki. Representing anaphora with dependent types. In Nicholas Asher and S. V. Soloviev, editors, *Logical Aspects of Computational Linguistics (LACL2014), LNCS 8535*, pages 14–29. Springer, 2014.

[3] Daisuke Bekki and Koji Mineshima. Context-passing and underspecification in Dependent Type Semantics. In *Modern Perspectives in Type Theoretical Semantics*, pages 11–41. Springer, 2017.

[4] Johan Bos, Elsbeth Mastenbroek, Scott Mcglashan, Sebastian Millies, and Manfred Pinkal. A compositional DRS-based formalism for NLP applications. In *In International Workshop on Computational Semantics*, pages 21–31, 1994.

[5] Robin Cooper. Type theory and semantics in flux. *Handbook of the Philosophy of Science*, 14:271–323, 2012.

[6] Philippe de Groote. Towards abstract categorial grammars. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics (ACL2001)*, pages 252–259, 2001.

[7] Philippe de Groote. Towards a Montagovian account of dynamics. *Proceedings of Semantics and Linguistic Theory XVI*, pages 1–16, 2006.

[8] Paul Dekker. Predicate logic with anaphora. *Semantics and Linguistic Theory*, 4:79–95, 1994.

[9] Tim Fernando. A type reduction from proof-conditional to dynamic semantics. *Journal of Philosophical Logic*, 30(2):121–153, 2001.

[10] Jeroen Groenendijk and Martin Stokhof. Dynamic predicate logic. *Linguistics and Philosophy*, 14(1):39–100, 1991.

[11] Jeroen Groenendijk, Martin Stokhof, and Frank Veltman. Coreference and modality. In *The Handbook of Contemporary Semantic Theory*, pages 179–216. Blackwell, 1996.

[12] Dag T. T. Haug. Partial dynamic semantics for anaphora: Compositionality without syntactic coindexation. *Journal of Semantics*, 31:274–294, 2013.

[13] Hans Kamp. A theory of truth and semantic representation. In *Formal Methods in the Study of Language*. Mathematical Centre Tract 135, 1981.

[14] Hans Kamp and Uwe Reyle. *From Discourse to Logic: Introduction to Model-theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Springer, 1993.

[15] Susanna Kuschert, Michael Kohlhase, and Manfred Pinkal. A Type-Theoretic Semantics for lambda-DRT. In *Proceedings of the Tenth Amsterdam Colloquium*, 1995.

[16] Zhaohui Luo. Formal semantics in modern type theories with coercive subtyping. *Linguistics and Philosophy*, 35(6):491–513, 2012.

[17] Scott Martin. *The dynamics of sense and implicature*. PhD thesis, Ohio State University, 2013.

[18] Scott Martin and Carl Pollard. A dynamic categorial grammar. In *Proceedings of Formal Grammar 2014*, pages 138–154. Springer, 2014.

[19] Per Martin-Löf. Intuitionistic type theory. *Bibliopolis*, 1984.

[20] Reinhard Muskens. Combining Montague semantics and discourse representation. *Linguistics and Philosophy*, 19:143–186, 1996.

[21] Reinhard Muskens. Language, lambdas, and logic. In Geert-Jan Kruijff and Richard Oehrle, editors, *Resource-Sensitivity, Binding and Anaphora*, pages 23–54. Springer, 2003.

[22] Rick Nouwen. On dependent pronouns and dynamic semantics. *Journal of Philosophical Logic*, 36(2):123–154, 2007.

[23] Aarne Ranta. *Type-theoretical grammar*. Oxford University Press, 1994.

[24] Mark J. Steedman. *The Syntactic Process*. The MIT Press, 2000.

[25] Göran Sundholm. Proof theory and meaning. In *Handbook of Philosophical Logic: Volume III: Alternatives in Classical Logic*, pages 471–506. Springer, 1986.

[26] Jan van Eijck. Incremental dynamics. *Journal of Logic, Language and Information*, 10(3):319–351, 2001.

[27] Jan van Eijck and Hans Kamp. Representing discourse in context. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*. MIT Press, 1997.

[28] Cees F. M. Vermeulen. Sequence semantics for dynamic predicate logic. *Journal of Logic, Language and Information*, 2(3):217–254, 1993.

[29] Roberto Zamparelli. Coordination. In Claudia Maienborn, Klaus von Heusinger, and Paul Portner, editors, *Semantics: An International Handbook of Natural Language Meaning*, pages 1713–1741. De Gruyter Mouton, 2011.

[30] Henk Zeevat. A compositional approach to discourse representation theory. *Linguistics and Philosophy*, 12:95–131, 1989.